PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G10L 3/00 — A1

(11) International Publication Number: WO 98/50907

(43) International Publication Date: 12 November 1998 (12.11.98)

(21) International Application Number: PCT/US98/09437

(22) International Filing Date: 6 May 1998 (06.05.98)

(30) Priority Data:
60/045,741 6 May 1997 (06.05.97) US

(71)(72) Applicants and Inventors: MARX, Matthew, T. [US/US]; 44 Pierce Avenue #1, Everett, MA 02149 (US). CARTER, Jerry, K. [US/US]; 38 Day Street #42, Somerville, MA 02144 (US). PHILLIPS, Michael, S. [US/US]; 653 Concord Avenue, Belmont, MA 02178 (US). HOLTHOUSE, Mark, A. [US/US]; 163 Upland Road, Newtonville, MA 02160 (US). SEABURY, Stephen, D. [US/US]; 52 Bay State Road, Boston, MA 02215 (US). ELIZONDO–CECENAS, Jose, L. [US/US]; One Emersion Place, 6–B, Boston, MA 02114 (US). PHANEUF, Brett, D. [US/US]; 380 Pine Street, Marsfield, MA 02050 (US).

(74) Agents: SUGIMURA, Audrey, M. et al.; McDermott, Will & Emery, 600 13th Street, N.W., Washington, DC 20005–3096 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

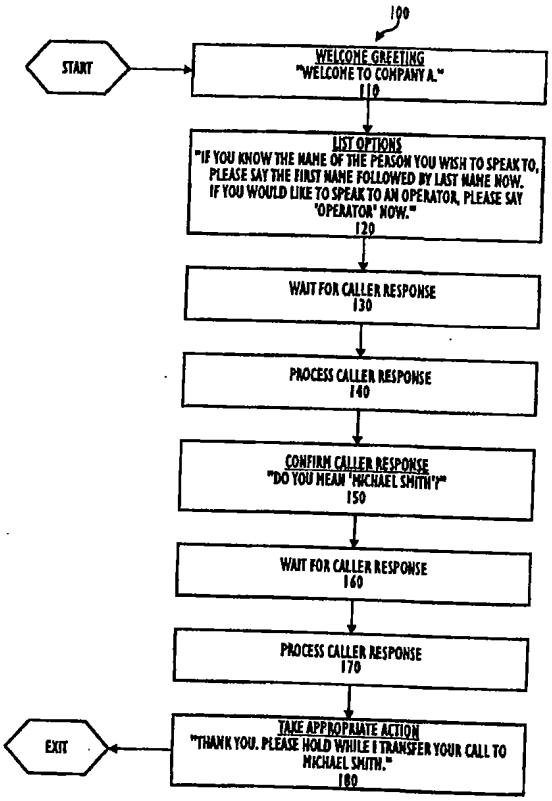
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SYSTEM AND METHOD FOR DEVELOPING INTERACTIVE SPEECH APPLICATIONS

(57) Abstract

The disclosed system and method for developing the invention store a plurality of dialogue modules in a speech processing system, wherein each dialogue module includes computer readable instructions for accomplishing a predefined interactive dialogue task in an interactive speech application. In response to user input (Figure 7, 51), a subset of the plurality of dialogue modules (Figure 7, 710, 720, 730) are selected to accomplish their respective interactive dialogue tasks and are interconnected in an order defining the call flow of the application (Figure 1, 110–180). A graphical user interface is disclosed, representing the stored plurality of dialogue modules as icons in a graphical display (Figure 7) in which icons are selected in the graphical display in response to user input, the icons for the subset of dialogue modules are graphically interconnected and the interactive speech application is generated based upon the graphical representation.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD FOR DEVELOPING INTERACTIVE SPEECH APPLICATIONS

Related Applications

This application claims priority from provisional application, U.S. Serial No. 60/045,741, filed on May 6, 1997, which is incorporated herein by reference.

Field of the Invention

The present invention relates generally to a system and method for developing computer-executed interactive speech applications.

Background

Computer-based interactive speech applications are designed to provide automated interactive communication, typically for use in telephone systems to answer incoming calls. Such applications can be designed to perform various tasks of ranging complexity including, for example, gathering information from callers, providing information to callers, and connecting callers with appropriate people within the telephone system. However, using past approaches, developing these applications has been difficult.

Figure 1 shows a call flow of an illustrative interactive speech application 100 for use by a Company A to direct an incoming call. Application 100 is executed by a voice processing unit or PBX in a telephone system. The call flow is activated when the system receives a incoming call, and begins by outputting a greeting, "Welcome to Company A" (110).

The application then lists available options to the caller (120). In this example, the application outputs an audible speech signal to the caller by, for example, playing a pre-recorded prompt or using a speech generator such as text-to-speech converter: "If you know the name of the person you wish to speak to, please say the first name followed by the last name now. If you would like to speak to an operator, please say 'Operator' now."

The application then waits for a response from the caller (130) and processes the response when received (140). If the caller says, for example, "Mike Smith," the application must be able to recognize what the caller said and determine whether there is a Mike Smith to whom it can transfer the call. Robust systems should recognize common variations and permutations of names. For example, the application of Figure 1 may identify members of a list of employees of Company A by their full names – for example, "Michael Smith."

However, the application should also recognize that a caller asking for "Mike Smith" (assuming there is only one employee listed that could match that name) should also be connected to the employee listed as "Michael Smith."

Assuming the application finds such a person, the application outputs a confirming prompt: "Do you mean 'Michael Smith'?" (150). The application once again waits to receive a response from the caller (160) and when received (170), takes appropriate action (180). In this example, if the caller responded "Yes," the application might say "Thank you. Please hold while I transfer your call to Michael Smith," before taking the appropriate steps to transfer the call.

Figure 2 shows some of the steps that are performed for each interactive step of the interactive application of Figure 1. Specifically, applying the process of Figure 2 to the first interaction of the application described in Figure 1, the interactive speech application outputs the prompt of step 120 of Figure 1 (210). The application then waits for the caller's response (220, 130). This step should be implemented not only to process a received response, as shown in the example of Figure 1 (140), but also to handle a lack of response. For example, if no response is received within a predetermined time, the application can be implemented to "time out" (230) and reprompt the caller (step 215) with an appropriate prompt such as "I'm sorry, I didn't hear your response. Please repeat your answer now," and return to waiting for the caller's response (220, 130).

When the application detects a response from the caller (240), step 140 of Figure 1 attempts to recognize the caller's speech, which typically involves recording the waveform of caller's speech, determining a phonetic representation for the speech waveform, and matching the phonetic representation with an entry in a database of recognized vocabulary. If the application cannot determine any hypothesis for a possible match (250), it reprompts the caller (215) and returns to waiting for the caller's response (220). Generally, the reprompt is varied at different points in the call flow of the application. For example, in contrast to the reprompt when no response is received during the time out interval, the reprompt when a caller's response is received but not matched with a recognized response may be "I'm sorry, I didn't understand your response. Please repeat the name of the person to whom you wish to speak, or say 'Operator.'"

If the application comes up with one or more hypotheses of what the caller said (260, 270), it determines a confidence parameter for each hypothesis, reflecting the likelihood that it is correct. Figure 2 shows that the interpretation step (280) may be applied for both low confidence and high confidence hypotheses. For example, if the confidence level falls within

a range determined to be "high" (step 260), an application may be implemented to perform the appropriate action (290, 180) without going through the confirmation process (150, 160, 170). Alternatively, an application can be implemented to use the confirmation process for both low and high confidence hypotheses. For example, the application of Figure 1 identifies the best hypothesis to the caller and asks whether it is correct.

If the application interprets the hypothesis to be incorrect (for example, if the caller responds "No" to the confirmation prompt of step 150), the application rejects the hypothesis and reprompts the caller to repeat his or her response (step 215). If the application interprets the hypothesis to be correct (for example, if the caller responds affirmatively to the verification prompt), the application accepts the hypothesis and takes appropriate action (290), which in the example of Figure 1, would be to output the prompt of 180 and transfer the caller to Michael Smith.

As exemplified by application 100 of Figures 1 and 2, interactive speech applications are complex. Implementing an interactive speech application such as that described with reference to Figures 1 and 2 using past application development tools requires a developer to design the entire call flow of the application, including defining vocabularies to be recognized by the application in response to each prompt of the application. In some cases, vocabulary implementation can require the use of an additional application such as a database application. In the past approaches, it has been time consuming and complicated for the developer to ensure compatibility between the interactive speech application and any external applications and data it accesses.

Furthermore, the developer must design the call flow to account for different types of responses for the same prompt in an application. In general, past approaches require that the developer define a language model of the language to be recognized, typically including grammar rules to generally define the language and to more specifically define the intended call flow of the interactive conversation to be carried on with callers. Such definition is tedious.

Because of the inevitable ambiguities and errors in understanding speech, an application developer also needs to provide error recovery capabilities, including error handling and error prevention, to gracefully handle speech ambiguities and errors without frustrating callers. This requires the application developer not only to provide as reliable a speech recognition system as possible, but also to design alternative methods for successfully eliciting and processing the desired information from callers. Such alternative methods may include designing helpful prompts to address specific situations and implementing different

methods for a caller to respond, such as allowing callers to spell their responses or input their responses using the keypad of a touch-tone phone. In past approaches, an application developer is required to manually prepare error handling, error prevention, and any alternative methods used in them. This is time consuming and may lead to omissions of functions or critical steps.

Based on the foregoing, there is a clear need in this field for an interactive speech development system and method that overcome these shortcomings.

Summary

In general, in one aspect, the invention features a computer-implemented method of constructing an interactive speech application by storing a plurality of dialogue modules in a speech processing system, wherein each dialogue module includes computer readable instructions for accomplishing a predefined interactive dialogue task in an interactive speech application. In response to user input, a subset of the plurality of dialogue modules are selected to accomplish their respective interactive dialogue tasks in the interactive speech application and are interconnected in an order defining the call flow of the application, and the application is generated.

Certain implementations may include one or more of the following features. The method may further include associating with specific dialogue modules, configuration parameters that change the operation of the dialogue module when the interactive speech program executes. The configuration parameters may be set in response to user input.

The interactive dialogue task associated with a dialogue module includes outputting a prompt to a caller and receiving a response from the caller. Examples of configuration parameters include: a timeout parameter defining a period for the caller to respond after a prompt is output; a prompt parameter defining a prompt to be output; an apology prompt parameter for defining an apology prompt to be output if the caller's response is not recognized; and a parameter for identifying a vocabulary defining recognizable responses from the caller. The method may further include editing the vocabulary in response to user input.

The interactive dialogue task associated with a dialogue module includes instructions for outputting a prompt to a caller, instructions for receiving a response from the caller, and instructions for interacting with a speech recognition engine for recognizing the received response using recognition models, and further may include instructions for updating the

recognition models used by the speech recognition engine based on recognized responses during execution of the interactive speech application.

The method further includes graphically representing the stored plurality of dialogue modules as icons in a graphical display, in which icons for the subset of dialogue modules are selected in the graphical display in response to user input, the icons for the subset of dialogue modules are graphically interconnected into a graphical representation of the call flow of the interactive speech application, and the interactive speech application is generated based upon the graphical representation. Using the graphical display, the method further includes associating configuration parameters with specific dialogue modules. Each configuration parameter causes a change in operation of the dialogue module when the interactive speech program executes. A window is displayed for setting the value of the configuration parameter in response to user input, when an icon for a dialogue module having an associated configuration parameter is selected.

In general, in another aspect, the invention features a memory device storing computer-readable instructions for constructing an interactive speech application in a speech processing system according to the method described above.

Among the advantages of the invention are one or more of the following. The present invention provides pre-packaged software modules each representing a discrete dialogue task for use in an interactive speech application. Because each of these "Dialogue Modules" performs a discrete task, they are largely independent, giving the application developer great flexibility in create a custom application by simply combining Dialogue Modules together in the order of the desired call flow of an application. In addition, because they are task-specific, Dialogue Modules can be optimized to provide the highest possible recognition accuracy and task completion rates through tuned semantic, language, and acoustic models.

By providing Dialogue Module templates in pre-packaged modules, the invention can be used to create applications that have internally consistent software code, which is especially important in larger applications with complex call flows.

Dialogue Module templates have customizable parameters, which provide developers with a high degree of flexibility in customizing an application. For example, although Dialogue Modules may be implemented to provide prerecorded "default" prompts to the caller, the developer may customize the prompts for specific applications. Other customizable features include whether to enable features such as a "barge-in" feature to recognize a caller's speech while the application prompt is running, selecting appropriate

error handling methods and prompts, and modifying or creating a database of recognized vocabulary.

The invention enables developers to create interactive speech applications to conduct automated conversations with callers, even if the developers have no formal speech training.

Brief Description of the Drawings

Fig. 1 is a flow chart of a call flow of an interactive speech application.

Fig. 2 is a flow chart of an interactive step in an interactive speech application.

Fig. 3 is a block diagram of a computer system on which interactive speech applications may be implemented.

Fig. 4 is a logical block diagram of a system suitable for developing interactive speech applications.

Fig. 5 is a flow chart illustrating an interactive speech application including Dialogue Module instances.

Fig. 6 is a flow chart of steps performed by a Dialogue Module.

Fig. 7 illustrates a Graphical User Interface (GUI) for creating and editing an interactive speech application.

Fig. 8 is a logical representation of an interactive speech application using Dialogue Modules.

Figs. 9-16 illustrate Graphical User Interfaces (GUIs) for creating and editing an interactive speech application.

Detailed Description

In the following description of a method and system for developing interactive speech application, for purposes of explanation, numerous specific details are set forth to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

I. Overview of a System for Developing Interactive Speech Applications

The invention is related to the use of a computer system for developing interactive speech applications. Figure 3 is a block diagram that illustrates such a computer system 300 upon which an embodiment of the invention may be implemented. Computer system 300

includes a bus 302 or other communication mechanism for communicating information, and a processor 304 coupled with bus 302 for processing information. Computer system 300 also includes a main memory 306, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 302 for storing information and instructions to be executed by processor 304. Main memory 306 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 304. Computer system 300 further includes a read only memory (ROM) 308 or other static storage device coupled to bus 302 for storing static information and instructions for processor 304. A storage device 310, such as a magnetic disk or optical disk, is provided and coupled to bus 302 for storing information and instructions.

Computer system 300 also includes output devices such as a display 312 coupled to bus 302 for displaying information to a computer user. Input devices coupled to bus 302 for communicating information and command selections to processor 304 may include a keyboard 314, a microphone 316, and a cursor control device 318, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 304 and for controlling cursor movement on display 312.

Computer system 300 also includes a communication interface 320 coupled to bus 302, providing coupling to external computer systems or networks. For example, as shown in Figure 3, communication interface 320 provides a two-way data communication coupling to a network link 322 that is connected to a local network 324. For example, communication interface 320 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection 322 to a corresponding type of telephone network lines 324. As other examples, communication interface 320 may be a telephony interface / voice board to provide voice and data communication connections 322 over telephone network lines 324, or may be a local area network (LAN) card to provide a data communication connection 322 to a compatible LAN 324. Wireless links may also be implemented. In any such implementation, communication interface 320 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 322 typically provides data communication through one or more networks to other data devices. For example, network link 322 may provide a connection through local network 324 to a host computer 326 or to data equipment operated by an Internet Service Provider (ISP) 328. ISP 328 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 330.

Additional details of exemplary components of a computer system suitable for speech systems are described in G. Pelton, "Voice Processing" (New York: McGraw-Hill, 1993), ISBN 0-07-049309-X, Chapter 8 ("Development Software").

According to one embodiment of the invention, an interactive speech application is developed and executed using software running on a general purpose computer system such as computer system 300. In alternative embodiments, special purpose hardware may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

Figure 4 is a functional block diagram of a system 400 for developing interactive speech applications. As used herein, a "Service" 410 is a customized interactive speech application developed to perform one or more dialogue tasks to provide a user-defined service. An example of a Service is the application described above with reference to Figures 1 and 2 for receiving and routing an incoming call to Company A.

An application developer creates a Service 410 defining a call flow in a runtime Service Execution Environment 420 which may be a default environment provided to the developer or a customized environment created or modified for the specific Service 410. In this embodiment, the Service Execution Environment 420 provides the "main( )" function that executes the Service 410, which is configured as a dynamically linked library (dll).

The call flow of the Service 410 describes its interactive conversation with callers using function calls to one or more "instances" of software Dialogue Modules from the Dialogue Modules 430. The system 400 comprises a plurality of Dialogue Modules, each designed for performing a specific dialogue task such as outputting a prompt, identifying the caller's speech as a recognized item of a predefined list, identifying the caller's speech as an affirmative or negative (Yes / No) response, or identifying strings of characters spelled by the caller. In the embodiments described herein, each Dialogue Module template is a function, method, object, or subroutine in a programming language such as C++, although a variety of different programming languages may be used.

A developer uses the Dialogue Modules to perform their respective dialogue tasks in a Service 410. Each Dialogue Module may use default configuration parameters or may be customized for specific Services. Parameters of a Dialogue Module instance may be customized to, for example, output customized prompts, recognize customized vocabularies in response to prompts, enable or disable specific features, and set a variety of additional parameters.

Dialogue Modules 430 provides an interface between the Service 410 and the Speech Components 440, 450, which perform functions enabling the system 400 to handle output and input audio signals. By providing the interface, the Dialogue Modules 430 allows a developer to develop a Service 410 without a detailed understanding of the Speech Components 440, 450, whose functions include outputting prompts to callers and receiving and processing input speech from callers. Any number of Speech Components 440, 450 may be included in the system 400.

In the embodiment illustrated in Figure 4, the Speech Output Components 440 output speech prompts (or other audio signals) through the Telephony Interface Components 460. In some cases, the Speech Output Components 440 may simply execute a specified audio file to output prerecorded speech. Alternatively, the Speech Output Components 440 may include a speech synthesis system, such as DECtalk™, a text-to-speech synthesizer that is available from Digital Equipment Corporation for converting text to speech. Commercially available speech synthesizers typically include a pronunciation dictionary and a speech generator to interpret an input text string, determine a pronunciation, and generate and output a speech waveform. Additionally, Speech Output Components 440 may include software to output an audio signal such as a beep when the prompt is finished playing, intended to notify callers that they should begin speaking. The Speech Output Components 440 may also include software to stop the output of a prompt if caller speech is detected to provide "barge-in" detection and handling. Details of barge-in detection and handling are explained, for example, in U.S. Serial No. 08/651,889, entitled "Method and Apparatus for Facilitating Speech Barge-In In Connection With Voice Recognition Systems," which is commonly assigned to the assignee of the present application.

The Speech Input Components 450 receive, record, and process incoming speech signals received through the Telephony Interface Components 460 during execution of a Service. Speech Input Components 450 typically include a Speech Recognition Engine such as that provided in SpeechWorks™, available from Applied Language Technologies, Inc. of Boston, Massachusetts, for recording, digitizing, and processing speech input. The Speech Recognition Engine, using additional components such as acoustic models for determining a phonetic representation of an input spoken word, database components for determining possible matches to entries in specified databases accessible to the Engine, and confidence correlation components for determining the confidence in hypotheses of possible matches, generates a textual representation of incoming speech signals received from callers. The

Engine has natural language modeling information, such as grammar rules of languages of speech it is intended to recognize.

The Telephony Interface Components 460 include components such as telephony cards providing telephony interface / voice boards for communicating over telephone lines, call channels for handling multiple calls on the telephone lines, an audio player / recorder for outputting prompts to callers and recording incoming speech from callers, and other components as needed to output and receive speech signals to and from callers, as well as software libraries to control the components.

The Service 410, including its Dialogue Module instances and their underlying Speech Components 440, 450 and Telephony Interface Components 460, operates within the runtime Service Execution Environment 420. As noted above, in this embodiment, the Service 410 is configured as a dynamically linked library (dll) and is executed by being called by the Service Execution Environment 420 which provides the "main( )" function. Additional software code is provided in a library to handle calls to Dialogue Module instances and other globally used functions.

In general, the Service Execution Environment 420 will invoke the Service 410 at three times: service initialization, service execution (processing incoming calls), and service clean up (after processing calls). Examples of functions the Service Execution Environment 420 can be configured to process include:

- initializing the telephony interface;
- initializing the interfaces to the Speech Components 440, 450 and Dialogue Modules 430;
- invoking user-provided service initialization routines, if necessary;
- waiting for an incoming call;
- determining a telephony channel;
- invoking user-provided service execution routines;
- ensuring disconnection of completed calls; and
- invoking cleanup routines, including user-provided service cleanup routines, Dialogue Module cleanup routines, and hardware/telephony resources cleanup routines.

II. Dialogue Modules

Interactive speech applications typically include a series of discrete dialogue tasks – requesting specific types of information from callers and processing the callers' responses. Dialogue Modules are predefined software components for performing these dialogue tasks

within an application. Each Dialogue Module performs a discrete task and saves its results, including a value indicating its termination condition. For example, termination conditions can include SUCCESS, indicating a successful completion of a dialogue task, TIMEOUT, indicating that the caller did not respond within a predefined timeout period, or ERROR, indicating that the system could not recognize the caller's response.

As noted above and illustrated in Figure 4, the Dialogue Modules 430 provide an interface between the Service 410 and Speech Components 440, 450, allowing developers to develop applications without a detailed understanding of speech technology. A Service 410 includes a series of calls to Dialogue Modules 430, ordered to produce the desired call flow and customized for the intended use of the specific Service 410. The Dialogue Modules 430, in turn, handle interaction with callers through the Speech Components 440, 450 and the Telephony Interface Components 460.

As used herein, Dialogue Module "templates" are predefined software modules that act as the building blocks for interactive speech applications, and Dialogue Module "instances" are versions of the templates as they are used in specific Services. A Dialogue Module instance may be identical to the template upon which it is based, or may be customized for a specific Service. Instances may be labeled with unique identifiers, allowing more than one instance of a Dialogue Module template to be used within a single Service.

Figure 5 is a flow diagram of an example of a Service 410 implemented using a system 400 such as that illustrated in Figure 4 and having a call flow as described above with reference to Figures 1 and 2. The Service 410 begins 510 by calling an ItemList Dialogue Module instance 520, whose task is to identify a person to whom a caller wishes to speak. The ItemList Module 520 begins by playing a prompt object 521, which in this example, uses the Speech Output Components 440 and Telephony Interface Components 460 to output a speech signal for the customized prompts shown by blocks 110 and 120 in Figure 1 and receiving the caller's voice response.

ItemList Module 520 accesses a customized recognized vocabulary having entries that identify people recognized by the Service 410. In the example of Figure 1, the recognized vocabulary corresponds to employees of Company A along with an operator and/or names of departments (e.g., sales, customer service, etc.). This customized vocabulary will typically be implemented by the application developer to recognize an employee not only by full name, but also by other names by which the employee could be recognized, such as just a last name, just a first name, or a nickname, perhaps combined with a last name. For example, if an employee is named Michael A. Smith, the database should recognize not only "Michael

Smith," but also other names by which a caller is likely to identify that employee, such as "Mike Smith," "Michael," "Mike," and "Smith." Such a vocabulary can be created using programs such as the Vocabulary Editor described below, or other appropriate data management applications.

In the confirmation step represented by block 523, the ItemList Module 520 identifies zero or more vocabulary entries as hypotheses of the person sought by the caller based on confidence levels determined by the Speech Input Components 450. In this embodiment, if the hypothesis having the highest confidence level has a confidence level exceeding a predefined threshold, the ItemList Module 520 assumes that hypothesis is the correct match for the caller's response. If no such hypothesis exists, the ItemList Module 520 determines hypotheses having confidence levels falling within a predefined range, indicating possible matches. The ItemList Module 520 sequentially outputs prompts for these hypotheses until a hypothesis is confirmed or the list of hypotheses is exhausted. More specifically, the confirmation step 523 receives and processes the caller's response to determine whether the response was affirmative or negative.

Processing the caller's response requires that the ItemList Module be able to understand and identify a variety of responses as affirmative or negative, including not only "yes" and "no", but also synonyms such as "correct," "incorrect," "right," "wrong," etc. Thus, the ItemList Module 520 also uses a recognized vocabulary for the confirmation step, having entries for recognized responses to the confirmation step, including information for each entry, indicating whether it represents an affirmative or a negative response.

Unlike the highly Service-specific recognized vocabulary used by the ItemList Module 520 to identify Company A employees, the recognized vocabulary of the confirmation step 523 is likely to be common in Services. Thus, the confirmation step 523 may be implemented to use a predefined default vocabulary that includes standard responses as entries. As explained below, however, if desired, the default database can be customized or replaced by the application developer for use with a specific Service. For example, in geographic areas with a Spanish-speaking population, Spanish vocabulary may be added to the database entries corresponding to affirmative responses.

If the ItemList Module 520 determines that the confirmation step 523 confirmed a hypothesis, it saves the hypothesis and Termination Condition (SUCCESS) and returns to the main function of the Service to transfer the call to the identified person 530. If the ItemList Module 520 determines that the confirmation step 523 does not confirm the hypothesis or terminated on a TIMEOUT or ERROR condition, the ItemList Module 520 may reattempt to

complete its task (repeating the cycle beginning with block 521, outputting a prompt and receiving and processing a response). Alternatively, the ItemList Module 520 can terminate on an ERROR condition 540 and take appropriate termination actions. In this example, a likely action would be to transfer the caller to a live operator.

Although not shown in Figure 1, a Dialogue Module may include alternative fallback methods for performing the dialogue task when it is unable to recognize or is unsure of the caller's response. Examples of such methods are asking the caller to spell his or her response, or to enter the response using the keyboard of a touchtone phone. In the Service 410 represented by Figure 5, the ItemList Dialogue Module instance 520 provides a spelling fallback 522 to callers.

After a predetermined number (which may be a default value, or a value set by the developer) of unsuccessful attempts to understand the caller during execution of the Service 410, the ItemList Module 520 uses a Spelling fallback method 522 to determine an entry in a vocabulary based on spelling received from the caller. In this example, the Spelling fallback method 522 uses the same recognition vocabulary used by the ItemList Module 520 and prompts the caller to spell the full name of the person to whom he or she wishes to speaks, first name followed by last name. The Spelling fallback method 522 searches the recognition vocabulary as it receives and converts each character of the caller's spelling.

The Spelling fallback method 522 is implemented to include a "look-ahead" feature, which is explained, for example, in copending application U.S. Serial No. 08/720,554, entitled "Method and Apparatus for Continuous Spelling Speech Recognition with Early Identification," which is commonly assigned to the assignee of the present application. Using the look-ahead feature, the Spelling fallback method 522 will terminate successfully as soon as it has identified a unique match between the characters spoken by the caller with an entry in the vocabulary, even if the caller has not completed spelling the entire word or phrase. If the Spelling fallback method 522 successfully identifies at least one entry, it saves the results and continues to the confirmation step 523, explained above.

.If the Spelling fallback method 522 does not identify a person, it saves a TIMEOUT or ERROR Termination Condition, as appropriate, and Exits 540. The action to take in case of error may be customized for various Services. As noted above, in the example of Figures 1 and 2, a likely termination action would be to transfer the caller to a live operator.

A. Common Features Of Dialogue Modules

The Dialogue Modules and the tasks they perform are of varying complexity and flexibility, ranging from simple, single-step modules to complex, multi-step modules. An example of a single-step module is a Yes / No Module that outputs a prompt and determines whether the caller's response is affirmative or negative. In contrast, an example of a multi-step module is one that requests an address from a caller. Such a module converts the caller's speech to text, and also correlates specific spoken words or phrases with specific fields of information, such as street name, city, state, and zip code.

While each Dialogue Module template handles a different dialogue task, the templates generally include common features to complete its dialogue task or exit gracefully. Figure 6 is a block diagram of some of these features, including prompting, collecting caller responses, optionally confirming caller responses, providing fallback methods for correctly recognizing caller responses, and disambiguating caller responses when necessary.

1. Initial Prompt

Most Dialogue Modules perform an interactive dialogue task that involves requesting and processing information from callers, and therefore include an initial prompt as shown in block 610, requesting the caller to speak the desired information. For example, the ItemList Module 530 of the Service described with reference to Figures 1, 2, and 5 uses an initial prompt that asks the caller to say the name of the person to whom the caller wishes to speak.

2. Collecting a Response

The collection step of block 620 is the second half of an interactive task – namely, receiving the caller's response. Dialogue Modules collect responses by recording waveforms of the caller's speech. To avoid causing the caller to wait indefinitely, this step generally has a "timeout" parameter that sets a predetermined time period for the caller to respond after the output of a prompt. Thus, there are two possible results: recognizing a received response, or not recognizing any response within the timeout period, as respectively indicated by control paths 620b and 620a.

Customizable features are provided in the collection step. For example, as explained below, the timeout period can be set to different lengths of time. Additionally, a beep-after-prompt feature can be enabled to output a beep (or any of a variety of other sounds) after a prompt is output when the timeout period begins (to signal the caller to speak). Similarly, a sound feature can be enabled to output sounds (such as percolation noises) after a caller has

finished speaking a response to let the caller know that the system is processing the response. Another feature, "barge-in" handling can be enabled to allow the collection step to detect and collect caller responses received before the preceding prompt has been completely output, and parameters such as the barge-in threshold can be set to determine when the barge-in features is used. As explained below, each of these parameters may be set by default values, or may be customized by the application developer.

3. Confirming the Response

If a response is received within the timeout period, a Dialogue Module attempts to confirm that it has correctly recognized the caller's response as shown by block 630. In this embodiment, the confirmation step 630 involves attempting to find a matching entry in a specified recognized vocabulary for the recorded waveform using the Input Speech Components 450. As noted above, in this embodiment, the Input Speech Components 450 include a Speech Recognition Engine that determines one or more hypotheses for a match and generates a score reflecting a confidence level for each hypothesis based on models such as semantic, language, and acoustic models. In one embodiment, the confirmation step 630 sequentially outputs confirmation prompts for each hypothesis, asking the caller to confirm whether a given hypothesis is correct, until a hypothesis is confirmed or until all of the hypotheses have been rejected.

The specified recognized vocabulary may be a default vocabulary or may be customized for a specific Service. For example, a Dialogue Module template such as the Yes / No Module may prompt the caller for responses that tend to be the same even in different Services. Such templates will typically be implemented to use a standard default vocabulary, although the vocabulary used by instances of these templates may be customized or changed by the developer. Other Dialogue Modules, such as the ItemList Module, will generally require customized vocabularies created for specific Services. As explained in greater detail below, a developer can create a customized vocabulary using an editing tool during development of a Service. Alternatively, an existing vocabulary can be dynamically updated during Service execution through a runtime application programming interface using technology such as that disclosed in copending application U.S. Serial No. 08/943,557, entitled "Method And Apparatus For Dynamic Adaptation Of A Large Vocabulary Speech Recognition System And For Us Of Constraints From A Database In A Large Vocabulary Speech Recognition System," which is commonly assigned to the assignee of the present application.

The confirmation step 630 uses a variety of factors to determine which, if any, of the vocabulary entries should be considered as hypotheses accurately representing the caller's response. Such factors include the confidence level generated by the Speech Recognition Engine of the Speech Input Components 450, a value for "n", the maximum number of hypotheses to consider (the n-best hypotheses), and prior recognition information.

More specifically, the confirmation step 630 will determine a confidence score for each vocabulary entry as a recognition hypothesis for a caller's response. As suggested in Figure 2, predefined threshold levels can be set to categorize confidence scores into high confidence, low confidence, or no confidence levels. These threshold levels can be varied, and determine which vocabulary entries the Recognition Engine will consider as hypotheses.

Another factor considered by the confirmation step 630 is the value for "n," which may be set to a default or customized value. The Recognition Engine uses this value to limit its consideration to the n-best hypotheses.

The Dialogue Module also retains information about prior recognition attempts for a given interactive session to adapt later recognition attempts to selectively screen out previously rejected hypotheses and to selectively consider repeated low-confidence hypotheses.

More specifically, if, during a first iteration of the cycle defined by blocks 610, 620, 630, and 640, the confirmation step 630 considers n-hypotheses, all of which are rejected as incorrect by the caller, in the next iteration, the confirmation step 630 will not consider any of those hypotheses again regardless of the confidence levels determined by the Recognition Engine. Conversely, if, during the first iteration of the cycle, the confirmation step 630 does not consider a hypothesis because the confidence level determined by the Recognition Engine falls within a low confidence interval (not high enough to be considered high confidence, but not low enough to be dismissed as "no hypothesis"), and that same hypothesis is identified again in the next iteration, the confirmation step 630 will consider that hypothesis.

Features that can be customized in Dialogue Module instances include disabling confirmation (in some cases, it is possible that a developer may choose to assume that the best hypothesis is always correct), enabling confirmation only under specified circumstances (such as if the confidence level for a given hypothesis is less than a predefined threshold), and enabling confirmation always. The prompt output by the confirming step may also be customized. For example, if a Dialogue Module instance is customized to determine the two best hypotheses for a caller's response, the confirmation step can be implemented to call a Yes / No Dialogue Module to determine whether the best hypothesis is correct, or may call a

Menu Dialogue Module (explained below), which lists both hypotheses and asks the caller to select one.

Other features that can be customized for the Confirmation Step 630 include the recognition vocabulary to use and a rejection threshold setting a minimum level of confidence for a vocabulary entry to be considered as one of the n-best hypotheses. Again, as explained below, each of these features can generally be provided by default parameters, or may be customized by the developer.

4. Disambiguating the Confirmed Response

The disambiguation step of block 660 disambiguates the caller's response if the confirmed response corresponds to more than one correct match with vocabulary entries. For example, in the Service illustrated in Figure 1, if the Company A ItemList vocabulary has two entries for employees named "Mike," the confirmation step 630 can determine when a caller to Company A asks to speak to "Mike" but cannot determine which of the two is requested by the caller. Thus, the Dialogue Modules also include a customizable disambiguation step 660 which, in this embodiment, outputs a customized prompt listing the possible entries and asking the caller to select one from the list.

5. Error Recovery

Dialogue Modules templates include error recovery methods which can be customized by a developer for specific instances in Services. For example, error recovery steps such as those represented by blocks 640 and 650 are executed after an unsuccessful attempt by a Dialogue Module instance to complete its dialogue task. For example, as shown in Figure 6, the error recovery steps 640, 650 are executed when the Service does not collect a response from the caller during the timeout period as shown by path 620a or the Service is unable to confirm any hypothesis matching the caller's response as shown by path 630a.

As explained below, the error recovery process is customizable for specific instances of Dialogue Modules in a Service. For example, at block 640, a Dialogue Module determines whether to reattempt to collect a response using the same method 610 as represented by path 640a, or using a fallback method, represented by path 640b to block 650. Error recovery parameters that can be customized include the content of prompts 610 used for reattempts following path 640a and threshold retry numbers to determine when to retry following path 640a and when to resort to the fallback method 650 following path 640b. For example, the retry numbers can include a maximum number of consecutive timeouts following a prompt

(following path 620a), a maximum number of consecutive errors in understanding a caller's responses to a specific prompt (following path 630a), and an overall maximum number of times the Dialogue Module instance will retry.

At the retry step of block 640, if the Dialogue Module determines that no threshold retry numbers have been reached, it will take path 640a to output another prompt to the caller at block 610. Common reprompts during a retry include two subcategories: apologies and reprompts. Apology prompts apologize for not completing its task, and may vary for different situations. For example, the apology prompt after a timeout may be, "I'm sorry, I didn't hear you," whereas the apology prompt after a recognition error may be, "I'm sorry for not understanding you." Similarly, the reprompts following the apology prompts may vary. The reprompt after a timeout apology prompt may say, "Please say your answer now," whereas the reprompt after a recognition error apology prompt may say, "Please repeat your answer now." Still other variations may be provided depending on the number and type of prior failures. For example, after a second consecutive timeout, the apology prompt may be, "I'm sorry, I still couldn't hear you," followed by the same reprompt, "Please say your answer now."

If the Dialogue Module determines that a threshold retry number is reached, it will follow path 640b to use a fallback method at block 650 to attempt to elicit a recognizable caller response. Examples of fallback methods include requesting the caller to spell his or her response or to enter DTMF tones using telephone touch-tone keys. For example, as illustrated in Figure 5, a Dialogue Module instance can be customized to exit to a Spelling Module 550 after a threshold number of recognition errors.

6. Termination

A Dialogue Module instance terminates either successfully at block 670 or unsuccessfully at block 680 and saves its Termination Condition. For example, Termination Conditions may include SUCCESS (for successful completion of the dialogue task), TIMEOUT (for expiration of a threshold number of time out periods), and ERROR (for unsuccessful attempts to recognize a caller's response).

In this embodiment, Dialogue Modules record information about the execution of the Dialogue Module instance at termination, including, for example, occurrences of the various execution steps such as collection, confirmation, disambiguation, starts and completions of prompts, and starts and completions of caller response recognitions. Recorded information

may include additional information such as the recorded waveforms of caller responses, timestamps, and the "n-best" recognition hypotheses and their confidence scores.

Logged recognition results and stored waveforms permit later analysis of the Service execution for uses such as trouble shooting, upgrading, and tuning. Additionally, such information can be used by Dialogue Module instances to improve completion rates by dynamically adjusting the semantic, language, and acoustic models used by the Speech Components 440, 450. These adjustments can be made at various levels. For example, adjustment can be made at a global level affecting the execution of Dialogue Module instances in all Services using the speech system. Adjustments can similarly be made at a caller level, affecting the execution Dialogue Module instances in Services interacting with a specified caller (identified by the Service Execution Environment using, for example, the calling telephone number, or by a Service using an identifier (such as an account number or user identification) for the caller).

Semantic adjustments are used by the Dialogue Modules to adapt recognition algorithms used by the Recognition Engine based on the meaning of a recognized caller response. For example, callers to an automated Service for making airline reservations typically request information regarding dates within a week or so of the calling date. A Dialogue Module instance for recognizing dates in such a Service can be implemented to consider the time interval within which dates recognized in prior calls fall, and to adjust the semantic models used by the Recognition Engine take into account that the date spoken by a caller in a future call is likely to fall within the same time interval.

Language adjustments are made to adapt recognition algorithms used by the Recognition Engine using information based on recognition results of prior executions of a Dialogue Module instance. For example, for a Dialogue Module instance for recognizing city names, such information may include tracking calling telephone numbers, callers' pronunciations of city names and the correctly recognized city names corresponding to those pronunciations in prior executions of the Dialogue Module instance, and the relative frequency with which certain words of the recognized vocabulary are spoken in comparison to other words. Such information can indicate that a response sounding like "Wuhstuh" from a caller from the Boston area (identifiable by the calling telephone number) is likely to mean the city Worcester, while the same response from a caller from the Providence area is more likely to mean the city Wooster. This information can be used by the Dialogue Module instance to adapt the language model used by the Recognition Engine. For example, if prior recognitions show that the response sounding like "Wuhstuh" from callers from the Boston

area are more likely to mean "Worcester" than "Wooster," the language model can be adapted to statistically weight the vocabulary entries "Worcester" and "Wooster" to indicate that "Worcester" is a more likely hypothesis than "Wooster" when the calling telephone number is from the Boston area.

Finally, acoustic adjustments can be made at the phoneme level to retrain the statistical acoustic models used by the Recognition Engine to associate specific sounds with the specific phonemes, based on information derived from the relationship between recognized phonemes and their corresponding pronunciations in received caller responses processed by the Dialogue Modules.

B. Examples of Dialogue Module Templates

Examples of individual Dialogue Modules (described with reference to Figure 6) include the following:

1. **Yes / No**: A Yes / No Module outputs an Initial Prompt and collects and determines whether a caller's response is affirmative or negative, based on a recognized vocabulary that includes a variety of responses as affirmative ("yes," "sure," "correct," "right," etc.) and as negative ("no", "incorrect," "wrong," etc.). The module saves a Termination Condition, and if successful, stores its result as, for example, a boolean value (0 for negative, 1 for affirmative).

The functionality of the Yes / No Module can be used within other Dialogue Modules. For example, as shown in Figure 5, the ItemList Dialogue Module may be implemented to provide the functions of the Yes / No Module in performing its Confirmation step 523.

2. **Spelling**: The Spelling Module outputs an Initial Prompt and collects one or more alphanumeric characters (including symbols) as the caller's response. In one embodiment, the module uses a specified vocabulary of entries to be recognized, and searches the vocabulary for matching entries as each character is spelled by the caller. As noted above, using the look-ahead feature provides early identification of spelled words and phrases. In an alternative embodiment, the module can use a specified vocabulary of individual characters, eventually identifying a character string corresponding to the spelling from the caller. Upon completion, the module saves a Termination Condition and if successful, stores its result in an appropriate format such as a data object or a character string.

As with the Yes / No Module, the functionality of the Spelling Module can be used within other Dialogue Modules. For example, as shown in Figure 5, the ItemList Dialogue

Module may be implemented to provide the functions of the Spelling Module in performing its fallback method 522.

3. **Formatted Codes**: A variety of module templates can be provided to recognize codes having specific formats, such as telephone numbers, addresses, zip codes, dates, times, currency amounts, account numbers, and social security numbers. These modules incorporate recognized grammars and use recognized vocabularies having entries corresponding to available response matching the format of the required code. Upon completion, the module returns a Termination Condition and if successful, stores its result as, for example, a data object or a character string.

4. **Menu**: The Menu Module outputs an initial prompt asking a caller to select from a series of listed options, collects the caller's response, and attempts to match the response with at least one entry of a recognized vocabulary corresponding to the listed options. Upon completion, the module returns a Termination Condition and if successful, stores its result as, for example, a data object or a character string.

5. **ItemList**: The ItemList Module lets a developer define a list of words or items as allowable responses to a caller prompt. In general, the initial prompt of this module does not constrain the caller's response (as does the Menu Module). For example, the ItemList Module in the Service of Figures 1 and 2 asks the caller to say the name of a person, without limiting the caller to specific responses. Such a module uses a recognized vocabulary including entries corresponding to recognized items. Upon completion, the module returns a Termination Condition and if successful, stores its result as, for example, a data object or a character string.

III. Creating A Customized Service

Referring again to Figure 4, a Service 410 may be implemented and integrated with Dialogue Modules 430 in several ways. For example, Service 410 can be configured as a "main" function in a third-generation programming language such as C, in which the Dialogue Modules 430 are called using C language function calls arranged in the call flow order. In this configuration, the "main" function can be a stand-alone program. Alternatively, as described above, the Service 410 may be configured as a dynamically linked library (dll) that is linked to a Service Execution Environment 420 when the Service 410 is initialized. In

this configuration, the Service 410 acts as a function library. The Service Execution Environment 420 executes one or more functions in the Service 410, which in turn call the Dialog Modules 430.

A. User Interfaces

Various user interfaces may be provided to allow different methods of creating the Service. For example, a non-graphic application programming interface may allow a developer to create the Service using traditional programming methods. Alternatively, graphical user interfaces (GUIs) may be used. For example, the GUI 700 illustrated in Figure 7 includes a stencil palette 710 having icons representing States (such as wait for call and transfer call) 720 and Dialogue Module templates 730, allowing the developer to structure the call flow of the Service by creating states and instances of Dialogue Modules by "dragging and dropping" the appropriate icons into the main workspace 740. The GUI 700 also includes a variety of connectors to link states and templates in the appropriate order and specifying appropriate conditions. In the embodiment shown, the stencil palette 710 is displayed along the left margin of the GUI window 700 and the various connector types are available in the pull-down menu 750. The Service icons are displayed in the main workspace 740 of the window.

To insert a state or Dialogue Module instance in the Service, a developer selects the appropriate icon from the stencil palette 710 and drops it into the main workspace 740, where it is displayed with the state or template name beneath the icon and the instance name above the icon. Initially, the GUI 700 automatically assigns a descriptive generic name to the instance, such as Instance #1. The developer can rename the instance by selecting and editing the text.

The menu bar 750 includes a "Connectors" option, which provides connector types to connect icons in the main workspace 740 in accordance with the desired callflow of the Service. Various connectors are provided. For example, an unconditional connector, represented in the main workspace 740 by a solid line, connecting a first and second icon indicates that the Service always proceeds to the second icon upon completion of the first. A conditional connector, represented in the main workspace 740 by a broken line, indicates that the Service proceeds to the second icon only if a condition is satisfied.

B. Customizing Dialogue Modules Instances in a Service

As explained above with reference to Figure 4, each Dialogue Module instance is based on a Dialogue Module template for accomplishing a discrete dialogue task and can be modified for specific Services. The relationship between a Service 410 and the Dialogue Modules 430, including both templates and instances, is illustrated in greater detail in Figure 8.

1. Dialogue Module Templates

The Dialogue Module Templates 810 include configuration libraries which define the behavior of the Dialogue Module instances 850 used in a Service 840. The configuration libraries include a "Baseline" configuration library 820 of default settings, including standard default parameters, prompt files, recognized vocabularies, header functions, and templates for the various dialogue tasks performed by the Dialogue Modules. A developer can customize the baseline configuration settings by providing an optional "System" configuration library 830, providing settings that override the default settings of the Baseline Library 820. The System Library 830 can provide override settings for any portion or all of the default settings.

2. Dialogue Module Instances in a Service

A developer can further customize a service by customizing the Dialogue Module Instances 850 in a Service 840. As noted above, each call within a service to a specified Dialogue Module is implemented by creating a separate "instance" of the corresponding Dialogue Module template and identifying each instance with a unique name. This enables Dialogue Module instances of the same template to be customized differently within a single Service. For example, as shown in Figure 8, the Service 840 makes two separate calls to Dialogue Module #2, represented by two separate instances 856A and 856B of the Dialogue Module templates 836, 826, which are based on the System Library settings 830 and any default Baseline Library settings 820 not overridden by the System Library 830.

The templates for the various Dialogue Modules share similarities, as illustrated by the above discussion with reference to Figure 6. These basic similarities allow customizable features common to multiple Dialogue Modules to be defined, including output prompts, parameters for recognition features such as the number of recognition candidates to consider, recognized vocabularies, and error recovery parameters.

Features can be customized during development or execution of a Service. For example, a developer can customize the features of a Service using text-based configuration

files, which allows the developer to change the parameters (and behavior during execution) of the Service without having to recompile the Service. Alternatively, features can be customized through runtime application programming interfaces contained within the Dialogue Modules. In embodiments integrated within a graphical development environment, a Graphical User Interface (GUI) can be provided to allow a developer to configure the Dialogue Modules by, for example, checking boxes or inserting text.

For example, using the GUI 700 of Figure 7, a Dialogue Module instance can be customized by selecting an icon such as the Directory Module 742 in the main workspace 740, which opens a dialogue window 900 such as that shown in Figure 9. This window 900 displays the name of the Dialogue Module instance 910 and provides four selections: Configuration Information 920, which allows the developer to view and modify the configuration information for the instance based on the information provided in the Baseline 820 and System 830 Libraries of the Dialogue Module Templates 810; Features 930, which allows the developer to customize various features for the instance; Vocabulary 940, which allows the developer to view, create, and edit recognized vocabulary for the instance; and Error Recovery 950, which allows the developer to view and modify the error recovery parameters for the instance.

a. Configuration Information

Selecting Configuration Information 920 from the window 900 of Figure 9 causes a new dialogue window 1000 such as that illustrated in Figure 10 to be displayed. Window 1000 displays the filepaths of the Baseline 820 and System 830 Libraries providing the configuration information for the Dialogue Module Instance 850. If more than one Baseline 820 and/or System 830 Library may be used, the window 1000 allows a developer to identify the desired Library in boxes 1010. The content of the configuration libraries may be viewed and or edited by selecting the View 1020 and Edit 1030 options.

b. Features

Selecting Features 930 from the window of Figure 9 opens a window such as that illustrated in Figure 11, displaying information about the various features that may be enabled in a specified Dialogue Module instance 850. The features shown in Figure 11 include an initial prompt, whether to enable "barge-in" handling, setting a barge-in threshold (how loud the caller must speak to enable barge-in handling), and whether to enable Beep After Prompt (playing a beep after the prompt to signal caller to speak). Parameters for these features are

initially set based on configuration information provided in the Baseline 820 and System 830 Libraries, but may be overridden by parameters entered by a developer in boxes 1110 - 1040.

Box 1110 is provided to specify the initial prompt for the Dialogue Module instance. As shown in Figure 6, a Dialogue Module generally includes one or more prompts to callers during its callflow. In general, prompts used by the Dialogue Module templates and instances are referred to as "prompt objects," which are data objects stored as either audio, or if used in a system having speech components that can synthesize speech from text, as text files. In this embodiment, the initial output prompt may be specified by the file path of a stored prompt object, or by text, to be converted upon execution by a text-to-speech synthesizer. Some Dialogue Module templates may provide a default initial prompt. For example, the Yes / No Module template might have a default initial prompt of "Please say yes or no." In other cases, a Dialogue Module template may require that the developer provide a custom initial prompt.

c. Vocabularies

Selecting the Vocabulary option 940 from the window 900 of Figure 9 allows a developer to customize the recognized vocabulary defining valid response to a Dialogue Module prompt. Some Dialogue Modules, such as the Yes / No Module, can use a completely defined default vocabulary, although such vocabularies may be customized or substituted by the developer. Other Dialogue Modules, such as the ItemList Module, while they may be used with a general standard vocabulary, are more suited for customized vocabularies.

In this embodiment, selecting the Vocabulary option 940 opens a window 1200 such as that shown in Figure 12. Figure 12 shows a vocabulary editor, which is a tool for customizing vocabularies for Dialogue Module instances. An example of an appropriate editor is the Vocabulary Editor, available in SpeechWorks™, commercially available from Applied Language Technologies, Inc. of Boston, Massachusetts. The SpeechWorks ™ Vocabulary Editor allows a developer to create or modify a recognized vocabulary, defining a list of terms that will be recognized in response to a prompt. The initial window 1200 shown in Figure 12 includes menu options to create a new vocabulary file 1210, open an existing file 1220, or exit 1230. For any opened file, the Editor provides three menu options: Items & Synonyms 1240, to edit the items and synonyms recognized; Pronunciations 1250, to edit pronunciations of items and synonyms recognized; and Confirmation 1260, to customize confirmation settings for the vocabulary file.

Figure 12 shows the display when the Items & Synonyms option 1240 is selected, providing options 1241-1245 for editing items and synonyms of the vocabulary. In this embodiment, "items" are the recognized items of a vocabulary, and "synonyms" of an item are alternative terms which the Dialogue Module will recognize as matches for that item. Figure 12 shows three items, "bob dole," "h ross perot," and "bill clinton."

Using a GUI such as that illustrated in Figure 12 provides option 1241 to allow a developer to add new terms as items, option 1242 to add synonyms for recognized terms, option 1243 to edit existing items and synonyms, option 1244 to delete existing items and synonyms, and option 1245 to insert another vocabulary file. Highlighting an existing item and adding a synonym allows the developer to add terms which the Dialogue Module instance should recognize as matching that item. For example, a synonym "clinton" could be added for the item "bill clinton." This could be displayed by listing synonyms under their items at an indented tier.

Selecting the Pronunciation option 1250 opens a window 1300 such as that shown in Figure 13. In the embodiment shown, the Vocabulary Editor determines pronunciations for items and synonyms using a predefined system dictionary that provides pronunciations of frequently used words in the language to be recognized, a user dictionary that includes user-defined pronunciations for specified words, and fall back rules, which are phonetic rules for generating pronunciations based on a word's spelling.

The window 1300 in Figure 13 shows each word 1310 of the items and synonyms from a vocabulary on the left, followed on the next line by a pronunciation 1320. In this embodiment the pronunciations are highlighted in a color reflecting its source. For example, a pronunciation from the system dictionary may be highlighted in white, a pronunciation from a user dictionary may be highlighted in light blue, a pronunciation edited by the user may be highlighted in darker blue, and a pronunciation generated from the phonetic rules may be highlighted in red.

As shown, the menu options 1251-1254 allow a developer to listen to the pronunciation of any of the items (or synonyms), edit the pronunciations, add alternative pronunciations, and delete pronunciations. Selecting a item (or synonym) followed by the Edit option 1252 or Add option 1253 opens a window 1400 such as that shown in Figure 14, displaying a phonetic keyboard 1410 and the selected entry 1420 for which a pronunciation is to be edited or added. Each key of the phonetic keyboard 1410 represents a phoneme. Holding the cursor over a key causes a pop-up box 1430 to open, providing an example of the pronunciation of the corresponding phoneme's sound by displaying a commonly used word

that includes that phoneme. The developer can also retrieve this information by selecting the "Show Table" option 1470, which will display the table of all available phonemes and a commonly used word illustrating their pronunciation.

To modify or add to the pronunciation for the selected item or synonym, a developer selects keys to insert the corresponding phonemes into the pronunciation. The Listen option 1440 allows a developer to hear the pronunciation of the phonemes shown in the Pronunciation box 1420 to aid the developer's verification and modification of a pronunciation.

Referring back to Figure 12, the Vocabulary Editor also provides a "Confirmation" option 1260. Selecting this option opens a window 1500 such as that illustrated in Figure 15. As noted above, Dialogue Module instances can be implemented to confirm its hypothesis of a caller's response. The Confirmation window 1500 of the Vocabulary Editor provides an option 1510 to allow a developer to set a default parameter for determining when items and synonyms from a specified vocabulary are confirmed, which is displayed in box 1511. The Confirmation window 1500 also provides an option 1520 to allow a developer to set confirmation parameters for individual items and synonyms of the vocabulary, which when selected, opens a window 1521 for the individual item or synonym, along with the available confirmation options.

d. Error Recovery Parameters

Selecting the Error Recovery option 950 from the window 900 of Figure 9 opens a window 1600 such as that illustrated in Figure 16, which allows a developer to customize the error recovery parameters to determine the call flow within a Dialogue Module instance. As noted above with reference to Figure 6 and as shown in Figure 16, error recovery parameters that can be customized include the timeout period, the maximum number of times a Dialogue Module will allow consecutive timeouts, the maximum number of times the Dialogue Module will allow consecutive recognition errors in understanding a caller's responses to a specific prompt, confirmation options, and fallback options. Default values for these parameters are initially provided by configuration information in the Baseline 820 and System 830 Libraries, and may be customized for specific instances of Dialogue Modules using the GUI such as the window 1600 of Figure 16.

Other error recovery parameters include the content of apology prompts and reprompts. Figure 8 shows a set of prompt files 822, 832 stored in the Baseline 820 and System 830 Libraries. Such files include files in the appropriate format for standard prompts

such as timeout apology prompts, error apology prompts, reprompts, and success messages. Customized prompts can also be provided in the prompt files, or stored elsewhere where they may be accessed within Dialogue Module instances.

As noted above, a variety of prompt may be provided in addition to the initial prompt, including, for example, a first and second timeout apology prompt, a first and second error apology prompt, as well as general reprompt prompts. The configuration data for Dialogue Module Templates 810 provided by the Baseline 820 and System 830 Libraries may include default prompts including a first timeout apology prompt of "I'm sorry, I didn't hear your response" and a second timeout apology prompt of "I'm sorry, I still couldn't hear your response," a first error apology prompt of "I'm sorry, I didn't understand what you said" and a second error apology prompt of "I'm sorry. I'm still having difficulty understanding you." The default prompts may also include a first general reprompt of "Please say your answer now," a second general reprompt of "Please say your answer again," and a default success prompt of "Thank you."

As noted above, the prompts may be specified in any appropriate format. For example, some embodiments may allow the prompt to be specified by its filepath, by a given name (for example, if named and stored in the Prompt Files 822, 832 shown in Figure 8), or if a text-to-speech synthesizer is used, by its text.

Some templates, such as the ItemList Module template, require the developer to create at least some of the prompts, using an appropriate Service to create and save the prompts so that they can be properly output to callers. For example to customize an existing prompt, a developer can open the prompt file in an appropriate Service and modify the prompt. To provide a new prompt, the developer can create a new prompt file and identify its filepath to Dialogue Module instances that output that prompt. Alternatively, in systems using text-to-speech synthesizers, a developer can simply provide the text of the prompt to a Dialogue Module instance.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A computer-implemented method of constructing an interactive speech application, comprising:

storing a plurality of dialogue modules in a speech processing system, wherein each dialogue module includes computer readable instructions for accomplishing a predefined interactive dialogue task in an interactive speech application;

selecting, in response to user input, a subset of the plurality of dialogue modules to accomplish their respective interactive dialogue tasks in an interactive speech application;

interconnecting, in response to user input, the selected subset of dialogue modules in an order defining a call flow of the interactive speech application; and

generating the interactive speech application.

2. The method of claim 1, further comprising:

associating at least one configuration parameter with at least one of the dialogue modules, wherein each configuration parameter causes a change in operation of the dialogue module when the interactive speech program executes; and

setting the value of the configuration parameter in response to user input.

3. The method of claim 2, wherein the interactive dialogue task associated with a dialogue module includes outputting a prompt to a caller and receiving a response from the caller, and one of the at least one configuration parameters is a timeout parameter defining a period for the caller to respond after a prompt is output.

4. The method of claim 2, wherein the interactive dialogue task associated with a dialogue module includes outputting a prompt to a caller and receiving a response from the caller, and one of the at least one configuration parameters is a prompt parameter defining a prompt to be output.

5. The method of claim 2, wherein the interactive dialogue task associated with a dialogue module includes outputting a prompt to a caller and receiving a response from the caller, and one of the at least one configuration parameters is an apology prompt parameter for defining an apology prompt to be output if the caller's response is not recognized.

6. The method of claim 2, wherein the interactive dialogue task associated with a dialogue module includes outputting a prompt to a caller and receiving a response from the caller, and one of the at least one configuration parameters is a parameter for identifying a vocabulary defining recognizable responses from the caller.

7. The method of claim 6, further comprising editing the vocabulary in response to user input.

8. The method of claim 1, wherein the interactive dialogue task associated with a dialogue module includes:

instructions for outputting a prompt to a caller;

instructions for receiving a response from the caller; and

instructions for interacting with a speech recognition engine for recognizing the received response using recognition models.

9. The method of claim 8, wherein the interactive dialogue task associated with a dialogue module further includes instructions for updating the recognition models used by the speech recognition engine based on recognized responses during execution of the interactive speech application.

10. The method of claim 1, further comprising:

graphically representing the stored plurality of dialogue modules as icons in a graphical display,

wherein:

icons for the subset of dialogue modules are selected in the graphical display in response to user input;

the icons for the subset of dialogue modules are graphically interconnected into a graphical representation of the call flow of the interactive speech application; and

the interactive speech application is generated based upon the graphical representation.

11. The method of claim 10, further comprising:

associating at least one configuration parameter with at least one of the dialogue modules, wherein each configuration parameter causes a change in operation of the dialogue module when the interactive speech program executes;

displaying a window for setting the value of the configuration parameter when an icon for a dialogue module having an associated configuration parameter is selected in response to user input; and

setting the value of the configuration parameter in response to user input.

12. A memory device storing computer-readable instructions for constructing an interactive speech application in a speech processing system, comprising:

sequences of instructions comprising a plurality of dialogue module templates, wherein each dialogue module template includes a sequence of instructions for performing a predefined interactive dialogue task in an interactive speech application;

instructions for creating, in response to user input, a plurality of dialogue module instances for use in an interactive speech application, wherein each dialogue module instance is based on one of the dialogue module templates and performs the predefined dialogue task of that corresponding dialogue module template in the interactive speech application;

instructions for customizing, in response to user input, at least one of the dialogue module instances;

instructions for interconnecting the dialogue module instances in an order defining a call flow of the interactive speech application; and

instructions for generating the interactive speech application.

13. The memory device of claim 12, further comprising:

instructions for associating at least one configuration parameter with at least one of the dialogue modules, wherein each configuration parameter causes a change in operation of the dialogue module when the interactive speech program executes; and

instructions for setting the value of the configuration parameter in response to user input.

14. The memory device of claim 13, wherein the interactive dialogue task associated with a dialogue module includes outputting a prompt to a caller and receiving a

response from the caller, and one of the at least one configuration parameters is a parameter for identifying a vocabulary defining recognizable responses from the caller.

15. The memory device of claim 14, further comprising instructions for editing the vocabulary in response to user input.

16. The memory device of claim 12, wherein the interactive dialogue task associated with a dialogue module includes:

instructions for outputting a prompt to a caller;

instructions for receiving a response from the caller; and

instructions for interacting with a speech recognition engine for recognizing the received response using recognition models.

17. The memory device of claim 16, wherein the interactive dialogue task associated with a dialogue module further includes instructions for updating the recognition models used by the speech recognition engine based on recognized responses during execution of the interactive speech application

18. The memory device of claim 12, further comprising:

instructions for graphically representing the stored plurality of dialogue modules as icons in a graphical display,

wherein:

the instructions for creating the plurality of dialogue module instances for use in an interactive speech application include instructions for selecting the plurality of dialogue module templates in response to user input and instructions for graphically representing the dialogue module instances as icons in the graphical display;

the instructions for interconnecting the dialogue module instances in an order defining a call flow of the interactive speech application include instructions for graphically interconnecting the icons representing the dialogue module instances into a graphical representation of the call flow of the interactive speech application; and

the instructions for generating the interactive speech application generate the interactive speech application based upon the interconnected icons graphically representing the dialogue module instances.

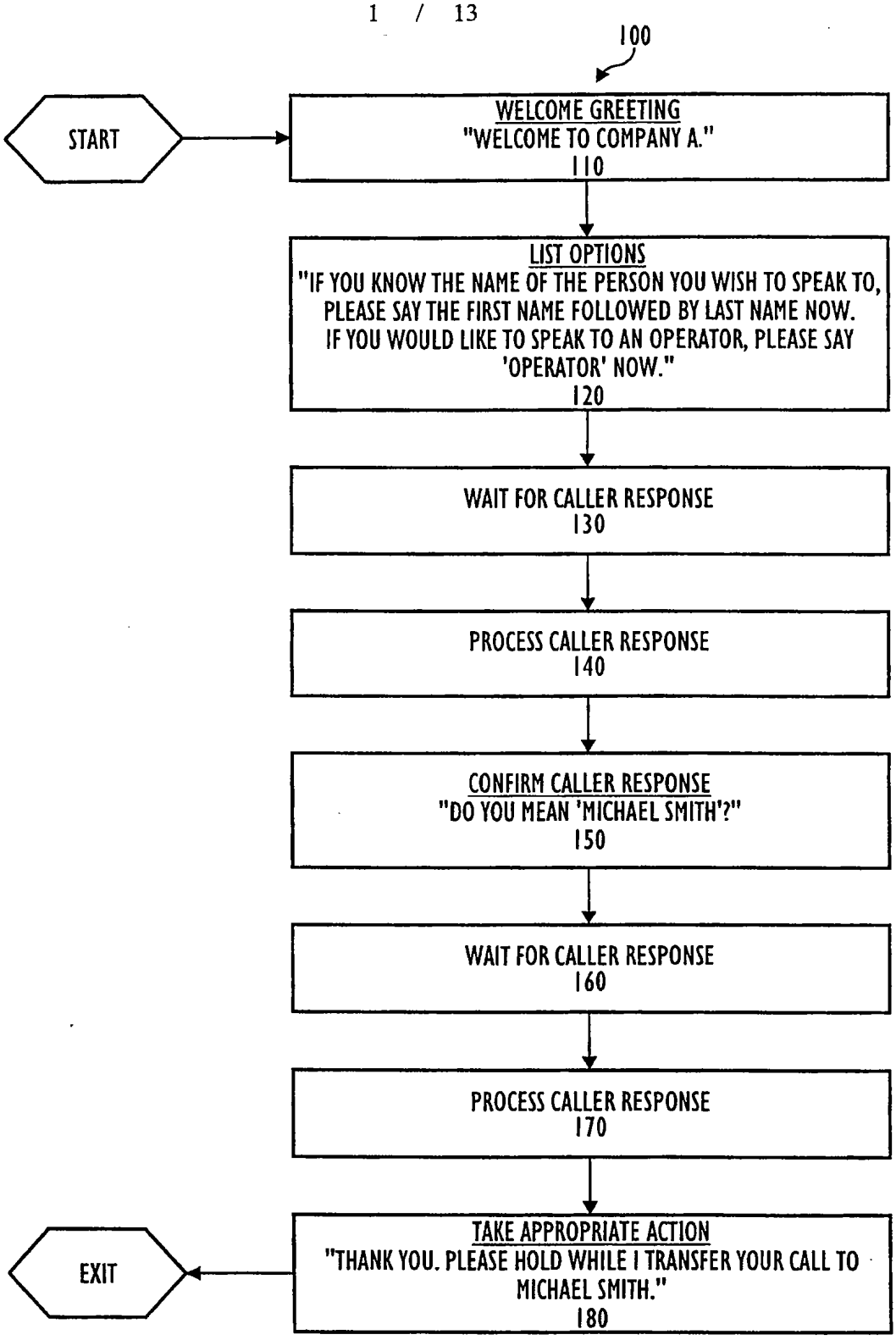

100
START
WELCOME GREETING
"WELCOME TO COMPANY A."
110
LIST OPTIONS
"IF YOU KNOW THE NAME OF THE PERSON YOU WISH TO SPEAK TO, PLEASE SAY THE FIRST NAME FOLLOWED BY LAST NAME NOW. IF YOU WOULD LIKE TO SPEAK TO AN OPERATOR, PLEASE SAY 'OPERATOR' NOW."
120
WAIT FOR CALLER RESPONSE
130
PROCESS CALLER RESPONSE
140
CONFIRM CALLER RESPONSE
"DO YOU MEAN 'MICHAEL SMITH'?"
150
WAIT FOR CALLER RESPONSE
160
PROCESS CALLER RESPONSE
170
TAKE APPROPRIATE ACTION
"THANK YOU. PLEASE HOLD WHILE I TRANSFER YOUR CALL TO MICHAEL SMITH."
180
EXIT

FIG. 1

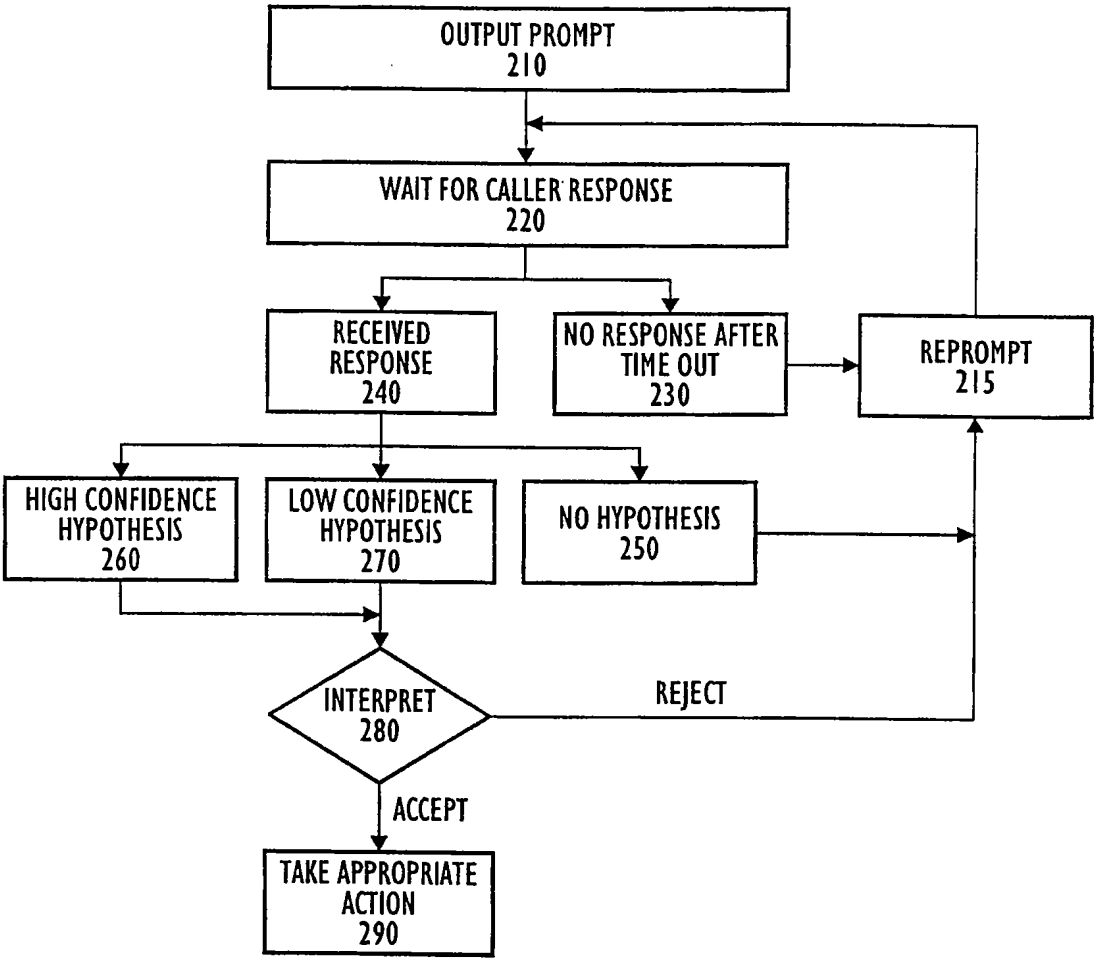

OUTPUT PROMPT
210
WAIT FOR CALLER RESPONSE
220
RECEIVED RESPONSE
240
NO RESPONSE AFTER TIME OUT
230
REPROMPT
215
HIGH CONFIDENCE HYPOTHESIS
260
LOW CONFIDENCE HYPOTHESIS
270
NO HYPOTHESIS
250
INTERPRET
280
REJECT
ACCEPT
TAKE APPROPRIATE ACTION
290

FIG. 2

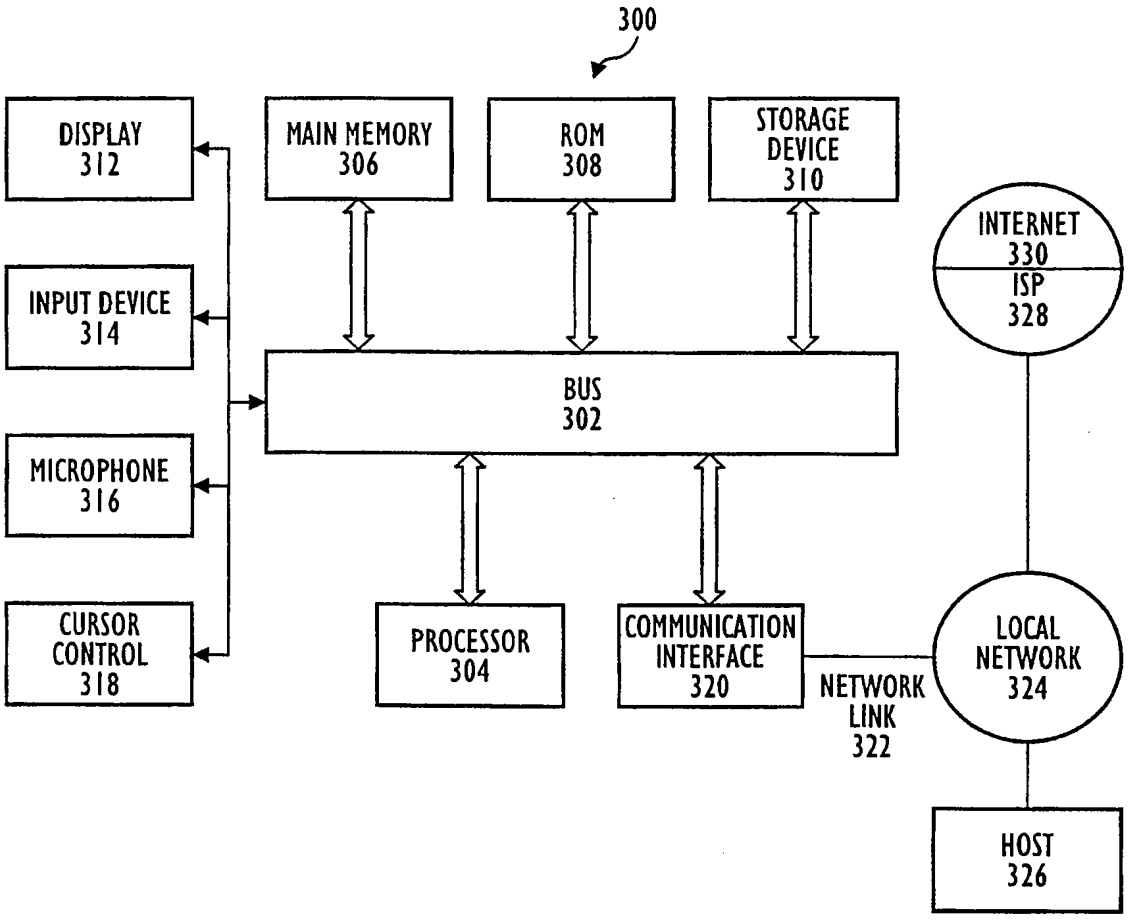

300
DISPLAY 312
MAIN MEMORY 306
ROM 308
STORAGE DEVICE 310
INTERNET 330
ISP 328
INPUT DEVICE 314
BUS 302
MICROPHONE 316
CURSOR CONTROL 318
PROCESSOR 304
COMMUNICATION INTERFACE 320
NETWORK LINK 322
LOCAL NETWORK 324
HOST 326

FIG. 3

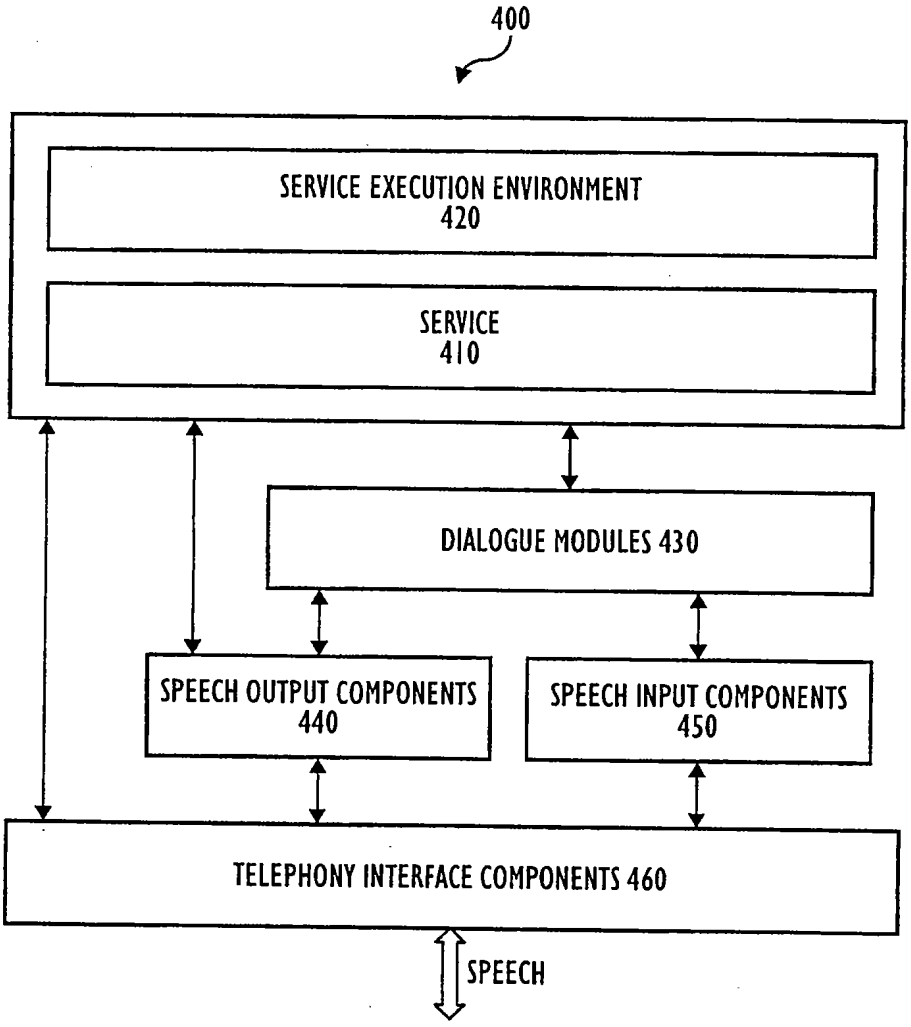

400
SERVICE EXECUTION ENVIRONMENT
420
SERVICE
410
DIALOGUE MODULES 430
SPEECH OUTPUT COMPONENTS
440
SPEECH INPUT COMPONENTS
450
TELEPHONY INTERFACE COMPONENTS 460
SPEECH

FIG. 4

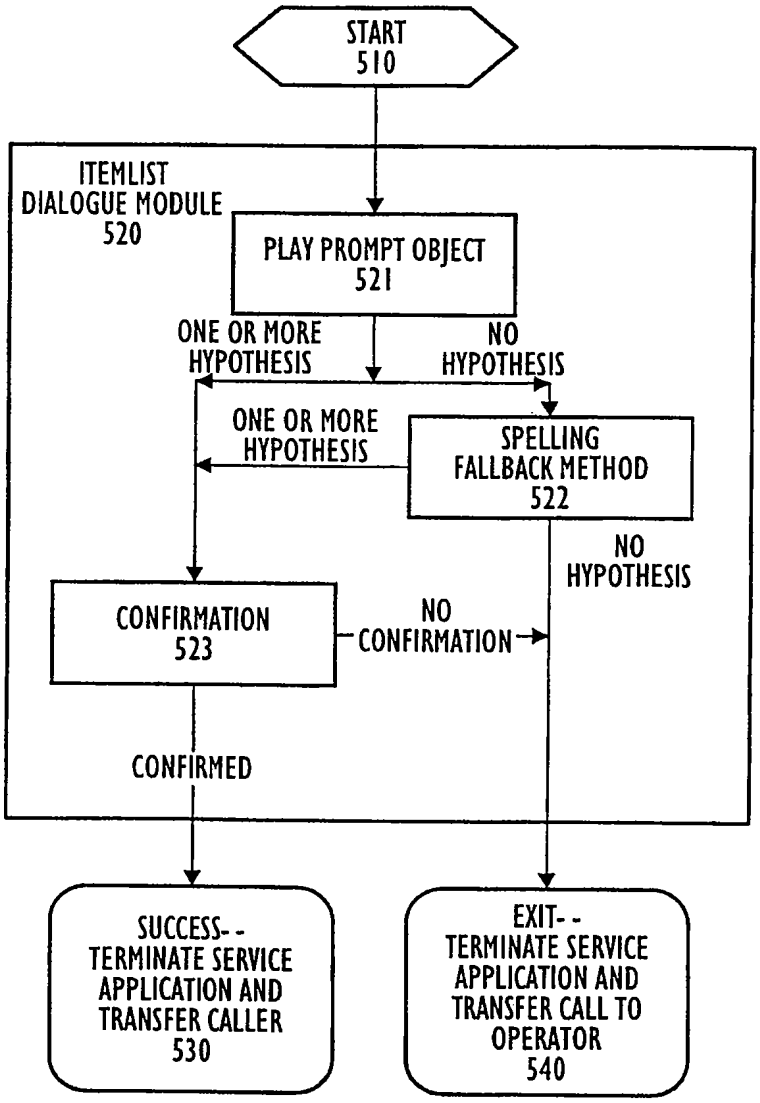

START
510
ITEMLIST
DIALOGUE MODULE
520
PLAY PROMPT OBJECT
521
ONE OR MORE
HYPOTHESIS
NO
HYPOTHESIS
ONE OR MORE
HYPOTHESIS
SPELLING
FALLBACK METHOD
522
NO
HYPOTHESIS
CONFIRMATION
523
NO
CONFIRMATION
CONFIRMED
SUCCESS- -
TERMINATE SERVICE
APPLICATION AND
TRANSFER CALLER
530
EXIT- -
TERMINATE SERVICE
APPLICATION AND
TRANSFER CALL TO
OPERATOR
540

FIG. 5

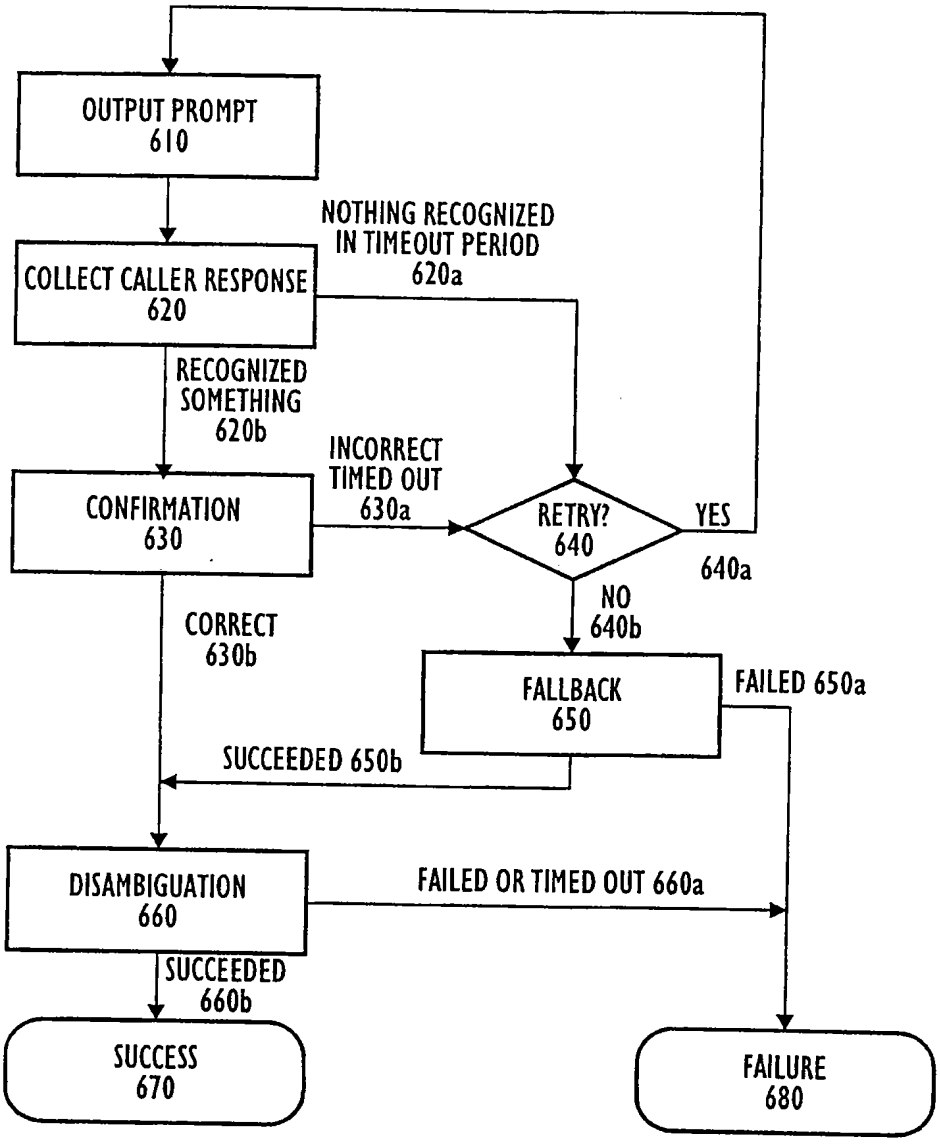

OUTPUT PROMPT
610
COLLECT CALLER RESPONSE
620
NOTHING RECOGNIZED
IN TIMEOUT PERIOD
620a
RECOGNIZED
SOMETHING
620b
CONFIRMATION
630
INCORRECT
TIMED OUT
630a
RETRY?
640
YES
640a
NO
640b
CORRECT
630b
FALLBACK
650
FAILED 650a
SUCCEEDED 650b
DISAMBIGUATION
660
FAILED OR TIMED OUT 660a
SUCCEEDED
660b
SUCCESS
670
FAILURE
680

FIG. 6

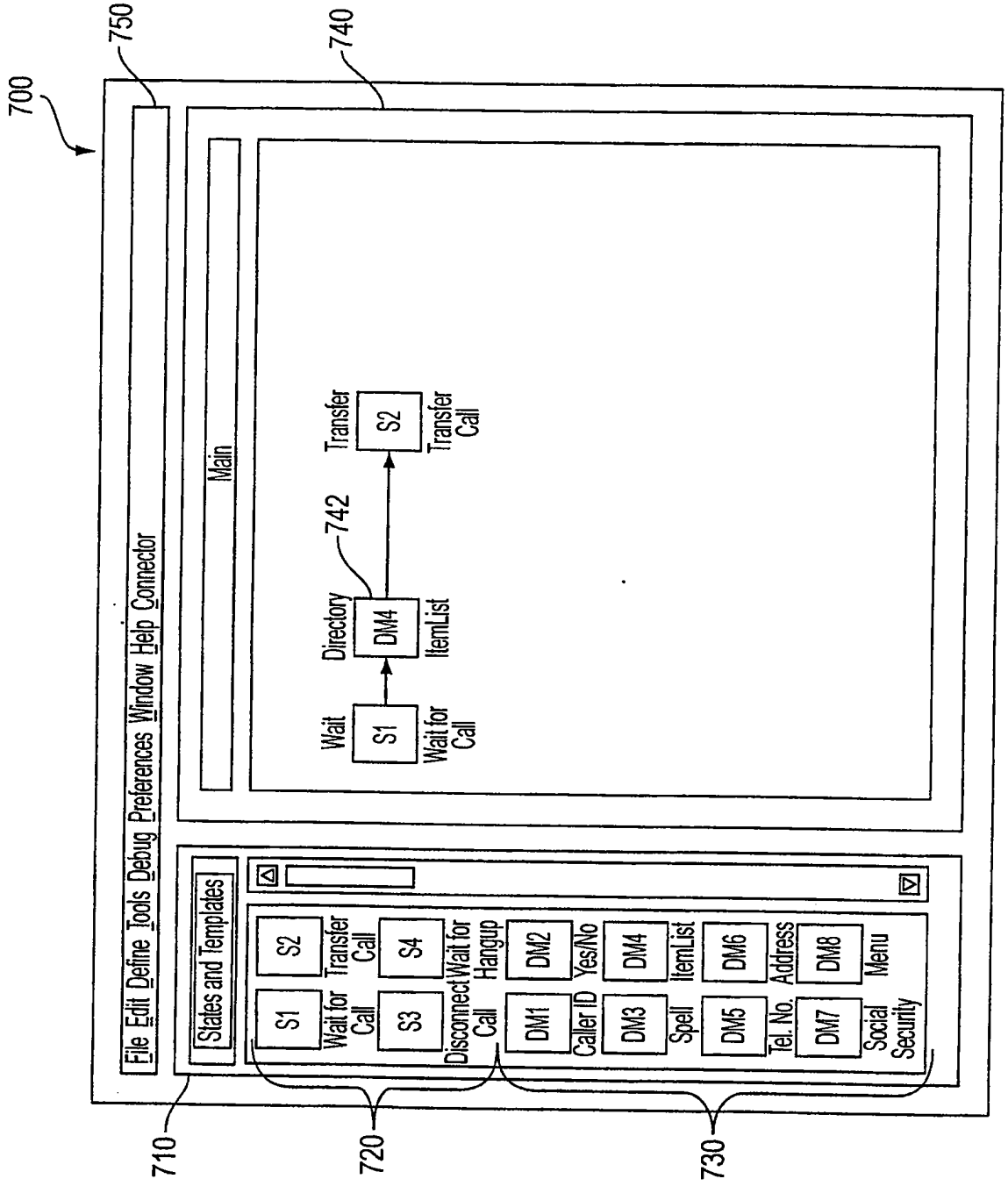

700
750
740
File Edit Define Tools Debug Preferences Window Help Connector
Main
Wait
S1
Wait for Call
Directory
DM4
ItemList
742
Transfer
S2
Transfer Call
States and Templates
S1
Wait for Call
S2
Transfer Call
S3
Disconnect
S4
Wait for Hangup
DM1
Caller ID
DM2
Yes/No
DM3
Spell
DM4
ItemList
DM5
Tel. No.
DM6
Address
DM7
Social Security
DM8
Menu
710
720
730

FIG. 7

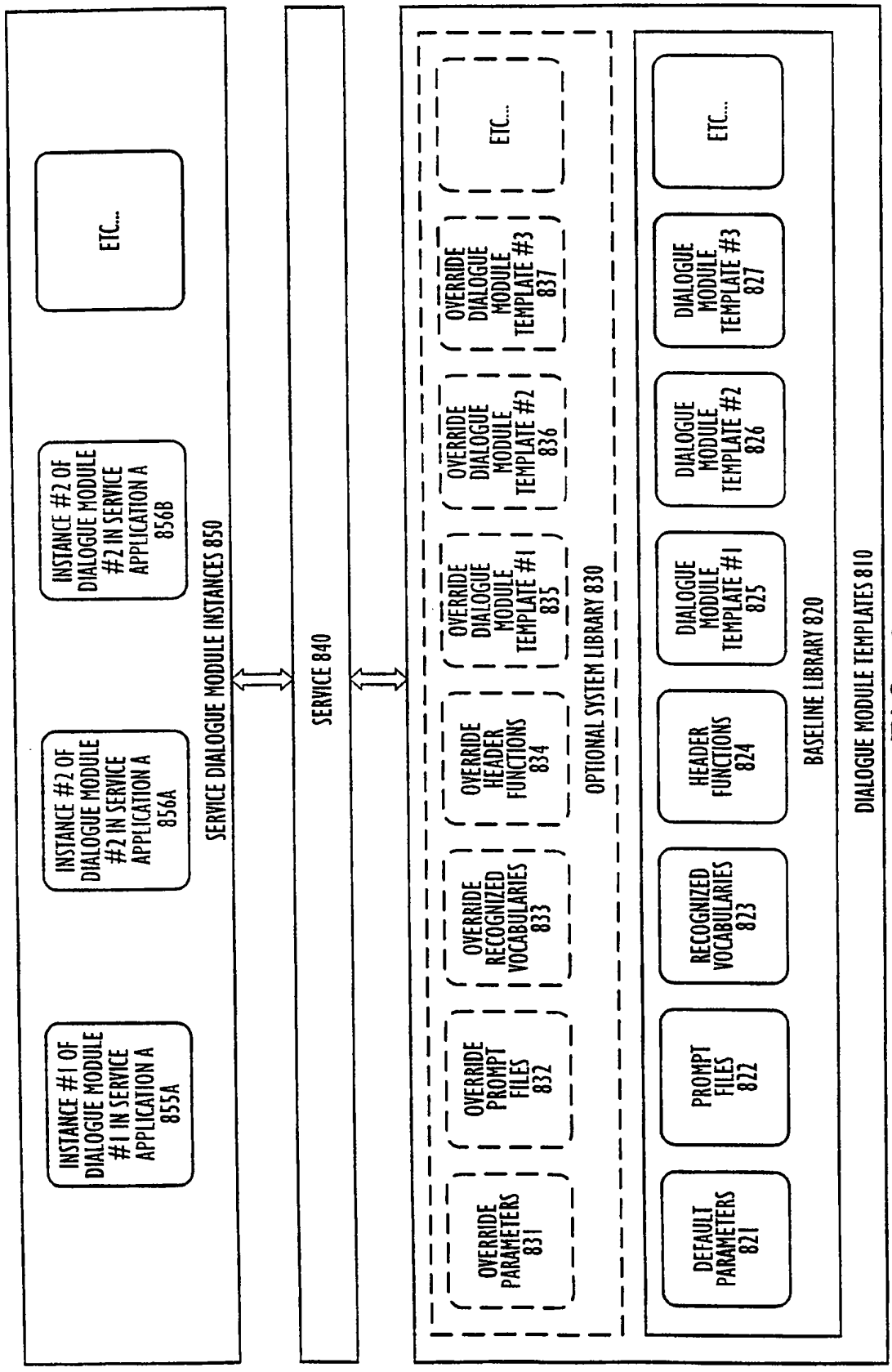

SERVICE DIALOGUE MODULE INSTANCES 850
INSTANCE #1 OF DIALOGUE MODULE #1 IN SERVICE APPLICATION A 855A
INSTANCE #2 OF DIALOGUE MODULE #2 IN SERVICE APPLICATION A 856A
INSTANCE #2 OF DIALOGUE MODULE #2 IN SERVICE APPLICATION A 856B
ETC...
SERVICE 840
DIALOGUE MODULE TEMPLATES 810
BASELINE LIBRARY 820
DEFAULT PARAMETERS 821
PROMPT FILES 822
RECOGNIZED VOCABULARIES 823
HEADER FUNCTIONS 824
DIALOGUE MODULE TEMPLATE #1 825
DIALOGUE MODULE TEMPLATE #2 826
DIALOGUE MODULE TEMPLATE #3 827
ETC...
OPTIONAL SYSTEM LIBRARY 830
OVERRIDE PARAMETERS 831
OVERRIDE PROMPT FILES 832
OVERRIDE RECOGNIZED VOCABULARIES 833
OVERRIDE HEADER FUNCTIONS 834
OVERRIDE DIALOGUE MODULE TEMPLATE #1 835
OVERRIDE DIALOGUE MODULE TEMPLATE #2 836
OVERRIDE DIALOGUE MODULE TEMPLATE #3 837
ETC...

FIG. 8

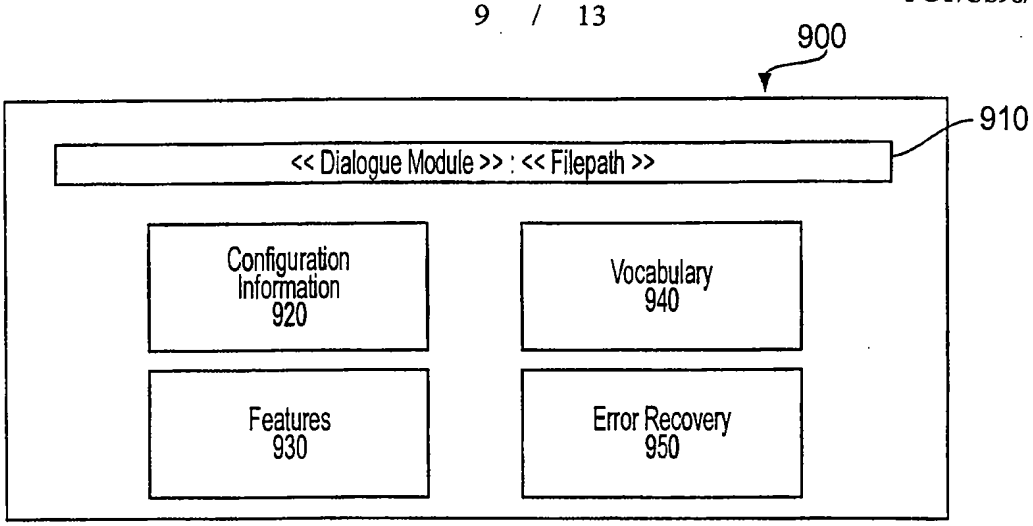


FIG. 9

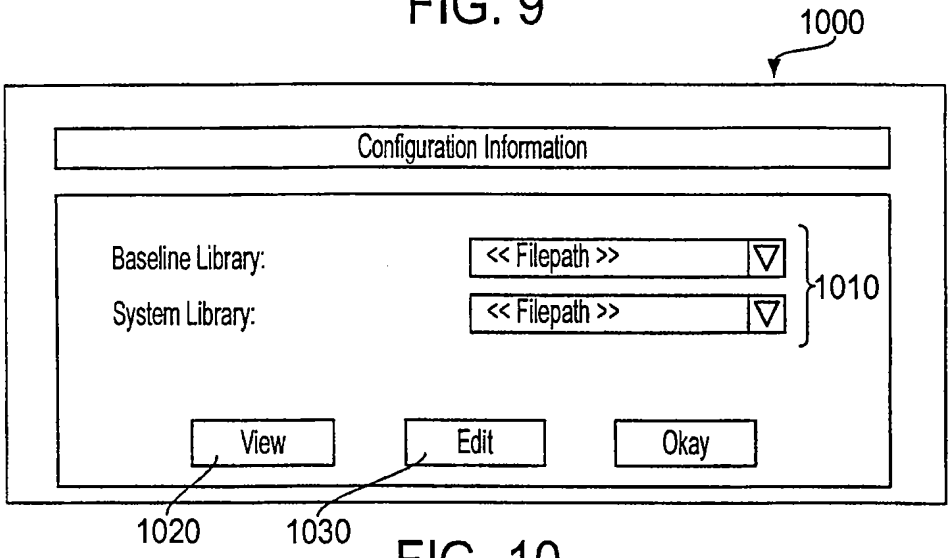


FIG. 10

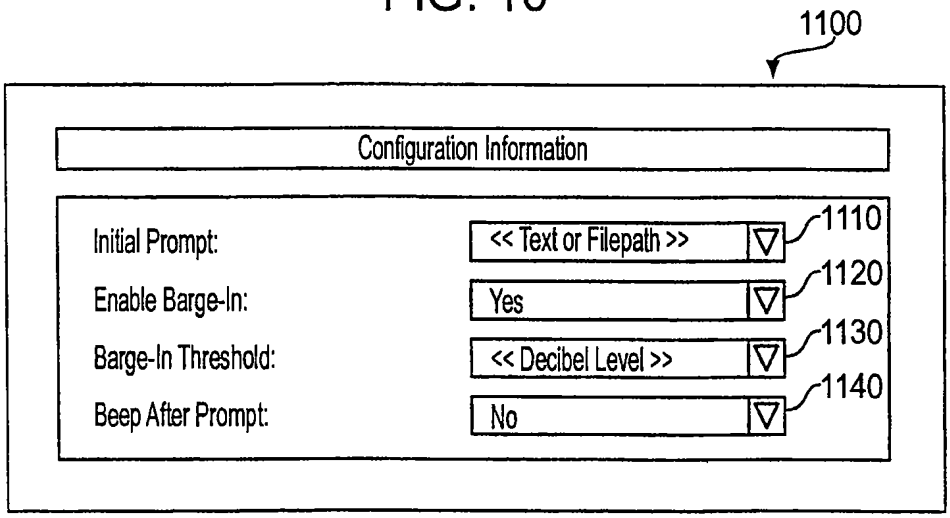


FIG. 11

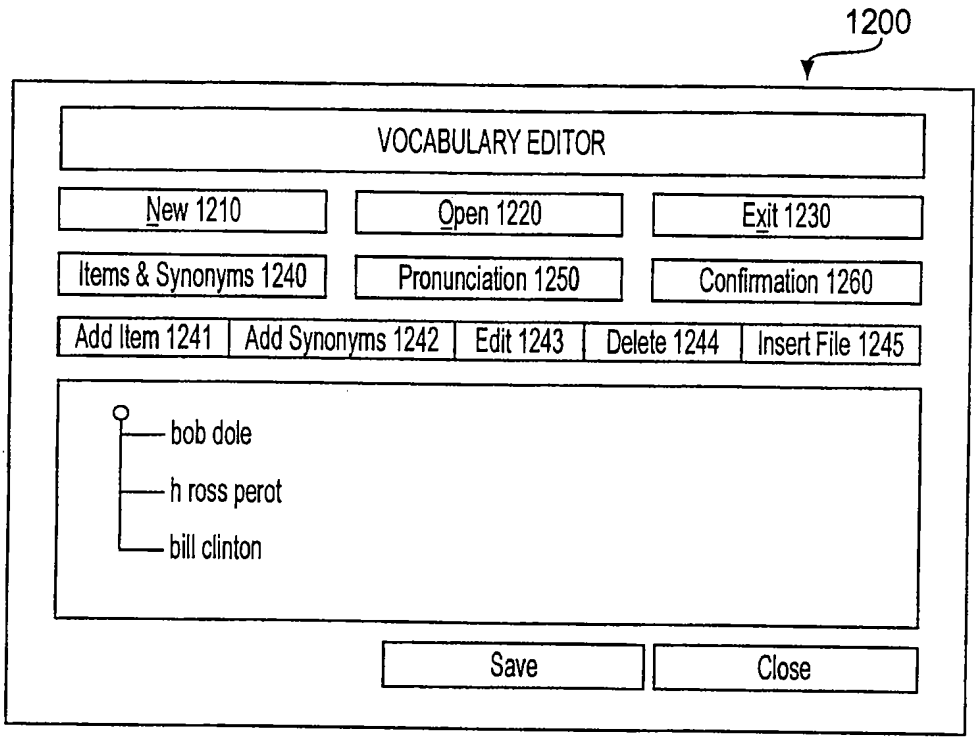

1200
VOCABULARY EDITOR
New 1210
Open 1220
Exit 1230
Items & Synonyms 1240
Pronunciation 1250
Confirmation 1260
Add Item 1241
Add Synonyms 1242
Edit 1243
Delete 1244
Insert File 1245
bob dole
h ross perot
bill clinton
Save
Close

FIG. 12

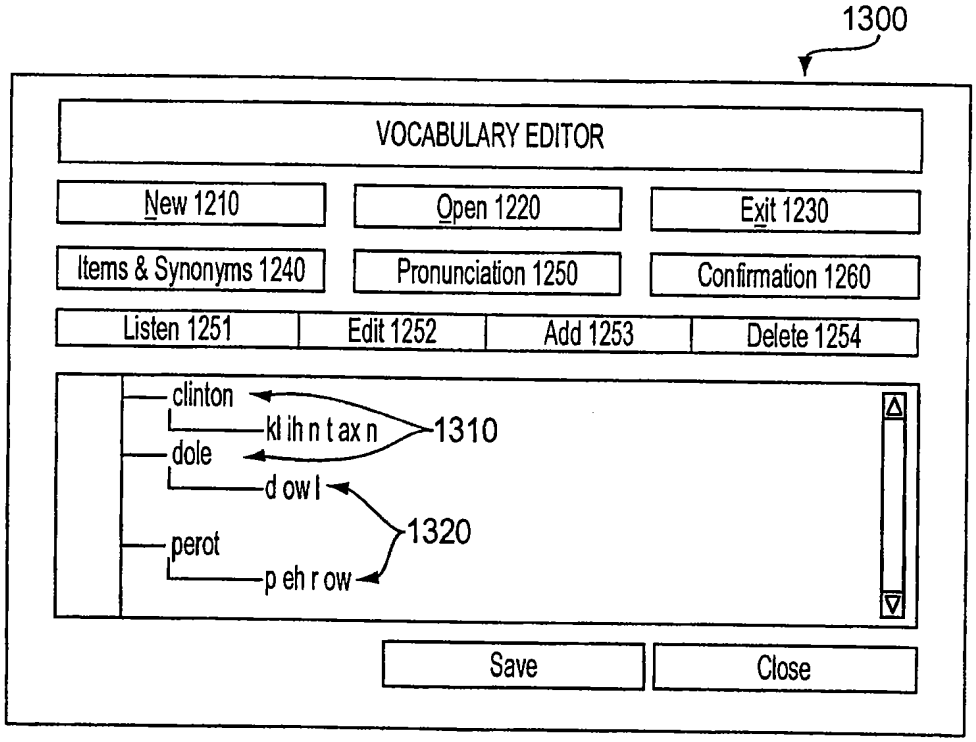

1300
VOCABULARY EDITOR
New 1210
Open 1220
Exit 1230
Items & Synonyms 1240
Pronunciation 1250
Confirmation 1260
Listen 1251
Edit 1252
Add 1253
Delete 1254
clinton
kl ih n t ax n
1310
dole
d ow l
1320
perot
p eh r ow
Save
Close

FIG. 13

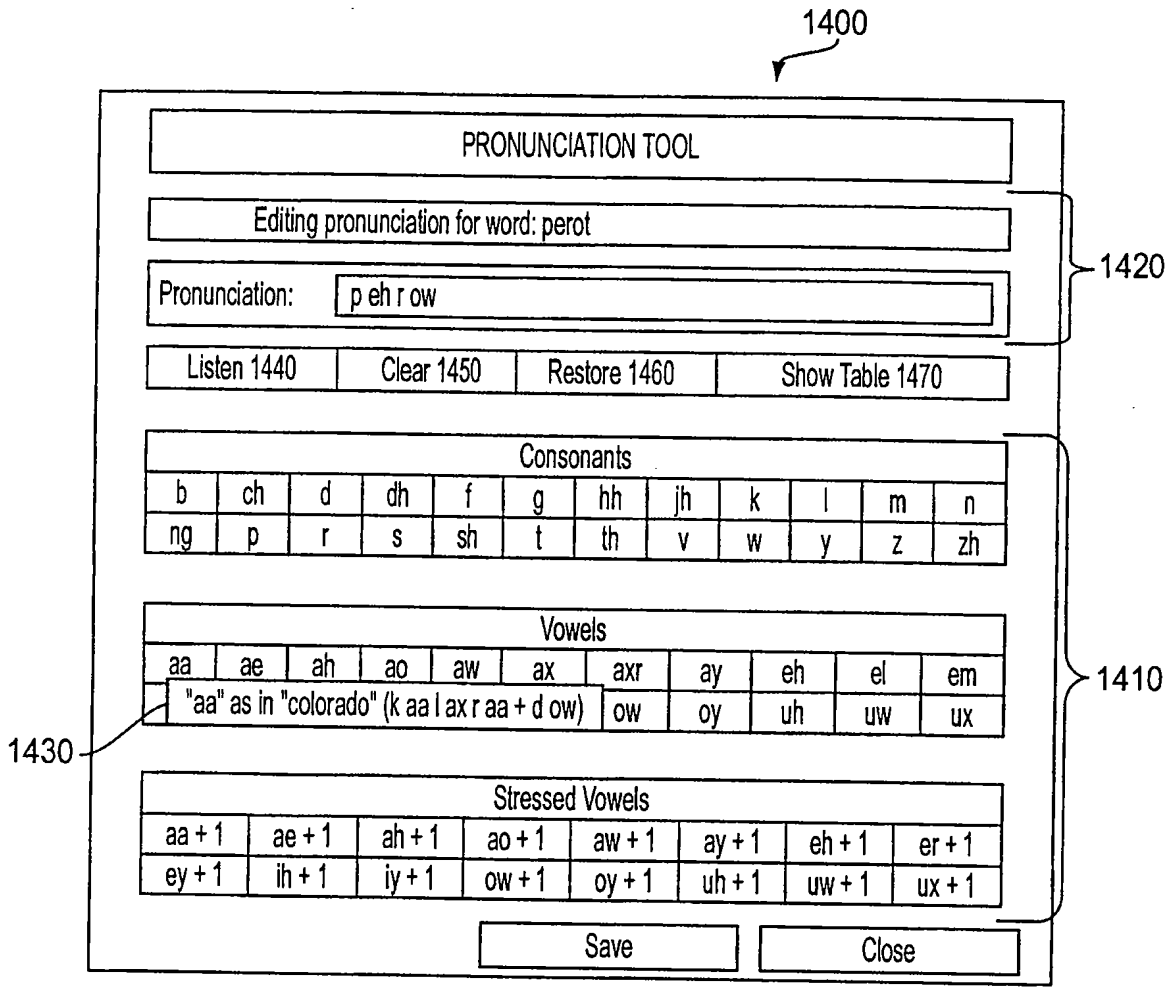

1400
PRONUNCIATION TOOL
Editing pronunciation for word: perot
Pronunciation: p eh r ow
1420
Listen 1440
Clear 1450
Restore 1460
Show Table 1470
Consonants
b ch d dh f g hh jh k l m n
ng p r s sh t th v w y z zh
Vowels
aa ae ah ao aw ax axr ay eh el em
"aa" as in "colorado" (k aa l ax r aa + d ow)
ow oy uh uw ux
1430
Stressed Vowels
aa + 1 ae + 1 ah + 1 ao + 1 aw + 1 ay + 1 eh + 1 er + 1
ey + 1 ih + 1 iy + 1 ow + 1 oy + 1 uh + 1 uw + 1 ux + 1
1410
Save
Close

FIG. 14

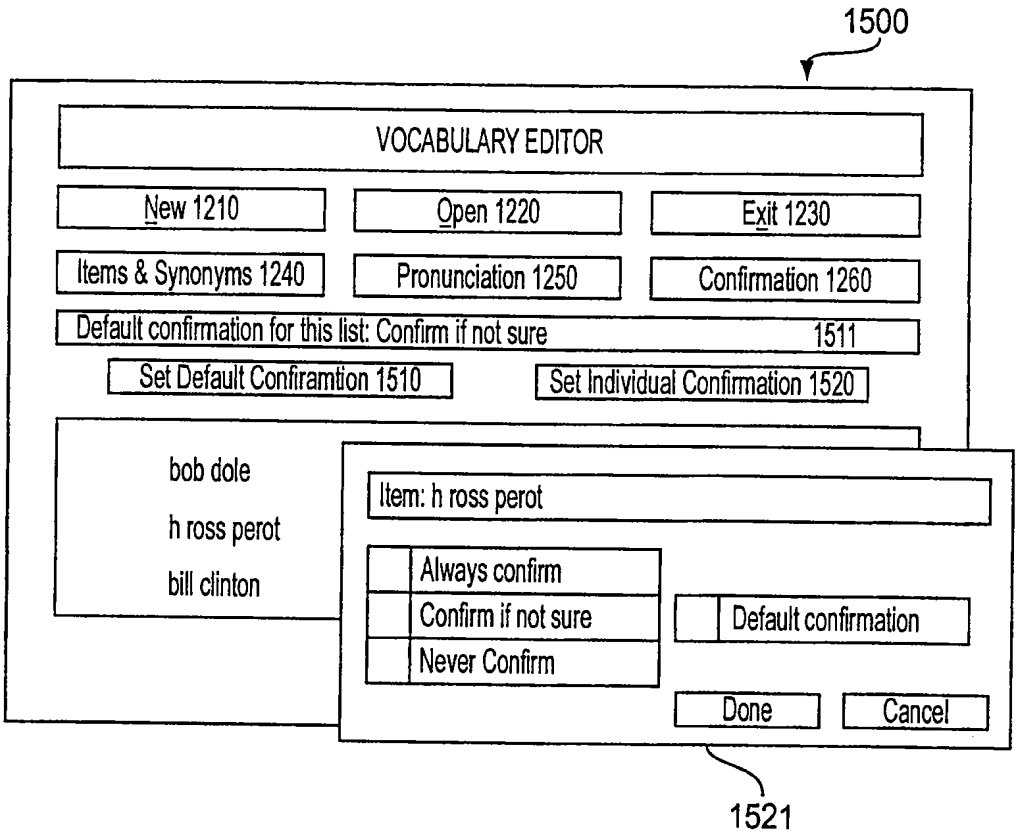

1500
VOCABULARY EDITOR
New 1210
Open 1220
Exit 1230
Items & Synonyms 1240
Pronunciation 1250
Confirmation 1260
Default confirmation for this list: Confirm if not sure
1511
Set Default Confiramtion 1510
Set Individual Confirmation 1520
bob dole
h ross perot
bill clinton
Item: h ross perot
Always confirm
Confirm if not sure
Never Confirm
Default confirmation
Done
Cancel
1521

FIG. 15

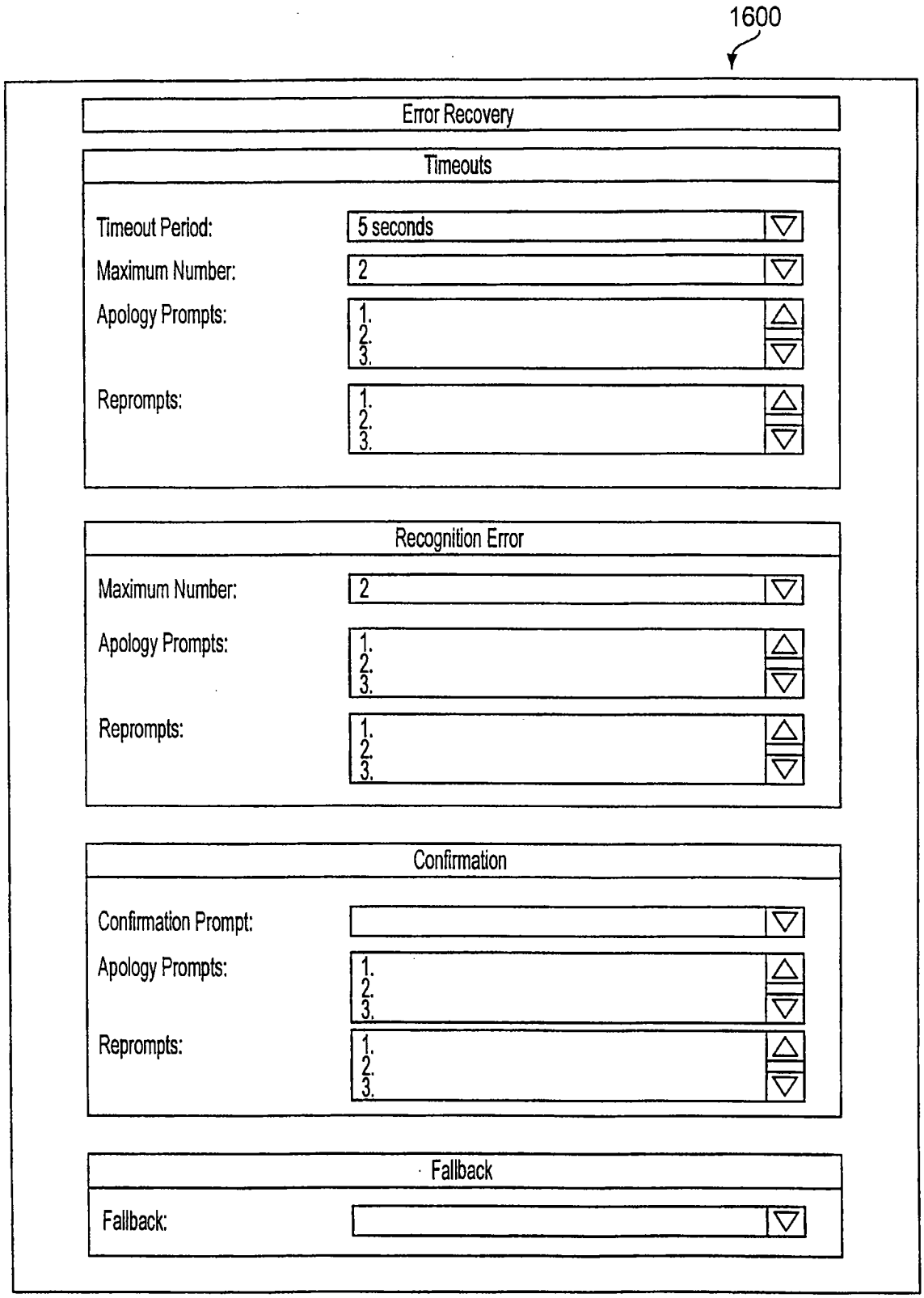

1600
Error Recovery
Timeouts
Timeout Period:
5 seconds
Maximum Number:
2
Apology Prompts:
1.
2.
3.
Reprompts:
1.
2.
3.
Recognition Error
Maximum Number:
2
Apology Prompts:
1.
2.
3.
Reprompts:
1.
2.
3.
Confirmation
Confirmation Prompt:
Apology Prompts:
1.
2.
3.
Reprompts:
1.
2.
3.
Fallback
Fallback:

FIG. 16

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/09437

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G10L 3/00
US CL :704/275,270,251,231
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 704/275,270,251,231

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, IEEE, MAYA

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,594,638 (Illiff) 14 January 1997, fig. 7,8,12-14,col. 28 -42	1-18
X,P	US 5,638,425 (Meador, III et al) 10 June 1997, figs. 5-7, col. 3 - col. 4, col. 6 line 39 - col. 10-col. 22	1-18
A	US 4,625,081 (Lotito et al) 25 Nov 1986, Fig. 1	1,12
A	US 5,774,860 (Bayya et al) 30 June 1998, fig. 1,4	1,12
A	US 5,652,789 (Miner et al) 29 July 1997, Figs. 1,3,6,10,	1,10,12,18

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 12 AUGUST 1998	Date of mailing of the international search report 06 OCT 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer DAVID R. HUDSPETH Telephone No. (703) 308-4825

Form PCT/ISA/210 (second sheet)(July 1992)★